Portland State University
PDXScholar

Dissertations and Theses | Dissertations and Theses

Spring 6-4-2019

Context-Aware Wi-Fi Infrastructure-based Indoor Positioning Systems

Huy Phuong Tran
Portland State University

Let us know how access to this document benefits you.

Follow this and additional works at: https://pdxscholar.library.pdx.edu/open_access_etds


Part of the Computer Sciences Commons

Recommended Citation
Tran, Huy Phuong, "Context-Aware Wi-Fi Infrastructure-based Indoor Positioning Systems" (2019). *Dissertations and Theses.* Paper 5009.

10.15760/etd.6885

This Dissertation is brought to you for free and open access. It has been accepted for inclusion in Dissertations and Theses by an authorized administrator of PDXScholar. For more information, please contact pdxscholar@pdx.edu.

Context-Aware Wi-Fi Infrastructure-based Indoor Positioning Systems

by

Huy Phuong Tran

A dissertation submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy
in
Computer Science

Dissertation Committee:
Nirupama Bulusu, Chair
Wu-chang Feng
Suresh Singh
Vivek Shandas

Portland State University
2019

© 2019 Huy Phuong Tran

Abstract

Large enterprises are often interested in tracking objects and people within buildings to improve resource allocation and occupant experience. Infrastructure-based indoor positioning systems (IIPS) can provide this service at *low-cost* by leveraging already deployed Wi-Fi infrastructure. Typically, IIPS perform localization and tracking of devices by measuring only Wi-Fi signals at wireless access points and do *not rely on inertial sensor data at mobile devices* (e.g., smartphones), which would require explicit user consent and sensing capabilities of the devices.

Despite these advantages, building an economically viable cost-effective IIPS that can accurately and simultaneously track many devices over very large buildings is difficult due to three main challenges. First, Wi-Fi signal measurements are extremely noisy due to unpredictable multipath propagation and signal attenuation. Second, as the IIPS obtain measurements in a best effort manner without requiring any applications installed on a tracked device, the measurements are temporally sparse and non-periodic, which makes it difficult to exploit historical measurements. Third, the cost-effective IIPS have limited computational resources, in turn limiting scalability in terms of the number of simultaneously tracked devices.

Prior approaches have narrowly focused on either improving the accuracy [33, 73, 79] or reducing the complexity [1, 79] of localization algorithms. To compute the location at the current time step, they typically use only the latest explicit Wi-Fi

measurements (e.g., signal strengths). The novelty of our approach lies in considering contexts of a device that can provide useful indications of the device's location. One such example of context is *device motion*. It indicates whether or not the device's location has changed. For a stationary device, the IIPS can either skip expensive device localization or aggregate noisy, temporally sparse location estimates to improve localization accuracy. Another example of context applicable to a moving device is a *floor map* that consists of pre-defined path segments that a user can take. The map can be leveraged to constrain noisy, temporally sparse location estimates on the paths.

The thesis of this dissertation is that embedding context-aware capabilities in the IIPS enhances its performance in tracking many devices simultaneously and accurately. Specifically, we develop motion detection and map matching to show the benefits of leveraging two critical contexts: *device motion* and *floor map*. Providing motion detection and map matching is non-trivial in the IIPS where we must rely only on data from the Wi-Fi infrastructure.

This thesis makes two contributions. First, we develop feature-based and deep learning-based motion detection models that exploit temporal patterns in Wi-Fi measurements across different access points to classify device motion in real time. Our extensive evaluations on datasets from real Wi-Fi deployments show that our motion detection models can detect device motion accurately. This, in turn, allows the IIPS to skip repeated location computation for stationary devices or improve the accuracy of localizing these devices. Second, we develop graph-based and image-based map matching models to exploit floor maps. The novelty of the graph-based approach lies in applying geometric and topological constraints to select which path segment to align the current location estimate. Our graph-based map matching can align a location estimate of a user device on the path taken by the user and close to the

user's current location. The novelty of the image-based approach lies in representing for the first time, input data including location estimates and the floor map as 2D images. This novel representation enables the design, development, and application of encoder-decoder neural networks to exploit spatial relationships in input images to potentially improve location accuracy. In our evaluation, we show that the image-based approach can improve location accuracy with large simulated datasets, compared to the graph-based approach. Together, these contributions enable improvement of the IIPS in its ability to accurately and simultaneously track many devices over large buildings.

To my parents and sister

for always supporting me

Acknowledgements

First and foremost, I would like to thank my advisor, Professor Nirupama Bulusu, for all of her advice and support. Seven years ago, Prof. Nirupama was the first person who introduced me to research when I was an undergraduate student at Portland State University. During the Ph.D. program, she has not only patiently strengthened my research skills but also given me the freedom to explore other fields that I am also interested in such as mobile health and robotics. She has given me the trust that she always does her best to guide and support her students. Over the years, I could not express in words how much I appreciate all of her support. I am proud to be one of her students.

I would like to thank the Connected Mobile Experiences team at Cisco Systems for all of their support during my internships over the past four years. I was deeply impressed the first time I saw how the team provided location-based services at many enterprises over the world. This motivated me to research in the field of indoor localization and tracking. In particular, I want to thank Santosh Pandey, Abhishek Mukherji, Rong Peng, and Abhishek Bhattacharyya for being my mentors. Furthermore, I would like to thank Xu Zhang and Oscar Bejarano Chavez from the Wireless AP team for valuable discussions. Overall, I would like to thank all of the team members. I feel that I was in a big family when I worked there.

I would like to thank Professor Wu-chang Feng, Professor Suresh Singh, and Pro-

fessor Vivek Shandas for serving on my dissertation committee. I appreciate all of their feedback.

I also feel grateful to have support from the staff at Portland State University. In particular, I would like to thank Rebecca Sexton for processing my paperwork and arranging rooms for my presentations. Also, I would like to thank Kristine-Anne Ronquillo Sarreal, Helen Frey, and Phyllis Bittinger for taking care of funding for buying lab equipment and conference travel.

I would like to thank Thanh Dang, who was my previous advisor at Washington State University Vancouver, for his guidance during my undergraduate research program, my master's program, and my first year in the Ph.D. program.

I want to thank my friends Hoang Le, Long Mai, Naif Alzahrani, Tien Le, Thong Doan, Nhut Le, and many more for sharing knowledge, joining my user studies and rehearsals as well as for other activities we have had together.

I want to thank my fiancee Linh Huynh for all of her support and encouragement. I feel so lucky to meet her.

Finally, I would like to acknowledge the grants and award that supported my research. This research was supported by Cisco Systems through grants CG 594206 and CG 1008109, and Maseeh College through Maseeh Fellowship award in 2016.

Table of Contents

List of Tables

List of Figures

1 Introduction

In the past four decades, Global Positioning System (GPS) receivers have become ubiquitous. Coupled with the astounding proliferation of mobile smartphones in the past decade, they have enabled transformative outdoor location-based services. Traveling outdoors has become much more convenient and cost-effective with many applications. Google Maps and Apple Maps allow people to perform navigation, track traffic conditions, and search nearby services in real time. Uber and Lyft have enabled location-based ridesharing. Online ordering from groceries and restaurants has become more convenient with applications such as Grubhub and Amazon that provide delivery time estimation and order tracking in real time. Furthermore, people can contribute to crowdsensing applications that report and monitor environmental conditions [53], parking spaces [11], and accessibility [46] in urban areas.

In indoor environments, knowing the spatial information of objects and occupants has the potential to similarly revolutionize indoor location-based services. While the full spectrum of possible services is yet to emerge, already many exciting scenarios are emerging. For instance, airports can significantly reduce waiting time of their customers at security lines by monitoring and forecasting the number of customers at these areas [58]. Hospitals can monitor the status of medical assets in real time to quickly locate available devices in urgent cases [3]. Retailers can advertise nearby sales information to customers based on their locations [41]. Buildings can also provide accessibility maps and routes. From a customer perspective, users within these large

commercial buildings can search for regions of interest and products as well as navigate within the premises quickly [5].

1.1 Infrastructure-based Indoor Positioning Systems (IIPS)

As GPS does not work indoors, several indoor positioning systems (IPS) have been proposed to perform object tracking [1] within buildings. Among IPS, infrastructure-based IPS (IIPS) that leverage already deployed Wi-Fi infrastructure within buildings to track objects are extremely popular [5, 27, 33, 73, 78]. Figure 1.1 gives a glimpse of the IIPS. Because the IIPS perform tracking of objects at location servers, the IIPS confer two important benefits: (i) supporting a wide range of devices including small form factor devices such as Wi-Fi tags that can be embedded into assets (ii) not requiring application installation on user devices such as phones. In Chapter 2, we describe important attributes of the IIPS in detail.

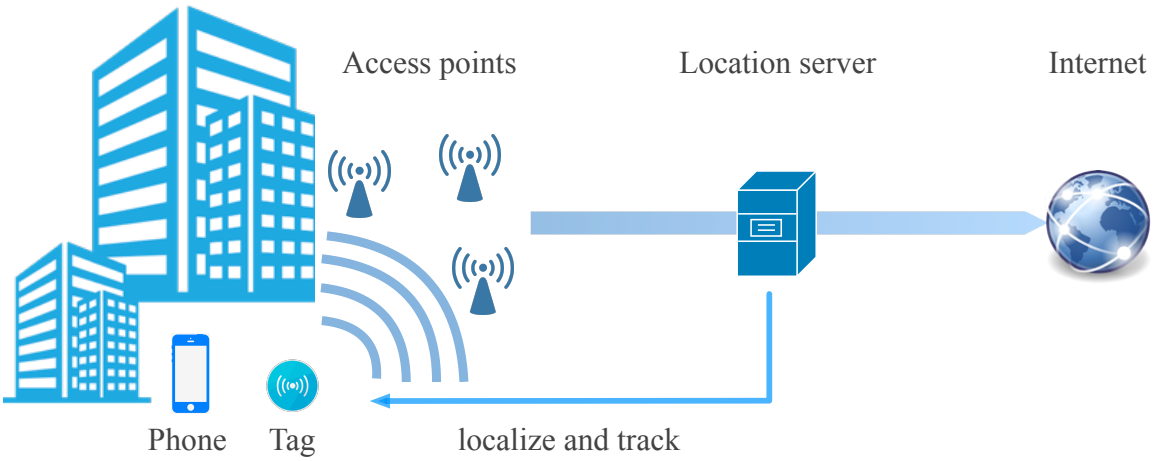


Figure 1.1: Illustration of Infrastructure-based Indoor Positioning Systems (IIPS). IIPS leverage existing Wi-Fi infrastructure including access points deployed within buildings to localize and track Wi-Fi devices.

[1] In this dissertation, we use tracking to refer to both localization and tracking.

1.2 The Problem: Scalable Accurate Localization & Tracking

The dominant goal of the IIPS is to enable various location-based services in large indoor environments such as hospitals, airports, and retail warehouses. To enable these services, the key requirement is that the IIPS need to localize and track thousands of devices *simultaneously* and *accurately*. Next, we describe the motivation for each of these two objectives.

The need for simultaneously tracking many thousands of devices is driven by two trends (i) the technological proliferation of Wi-Fi devices such as smartphones and tags which can be embedded in many assets and (ii) the application drivers for providing location-based services in large areas. Thus, the number of devices that the IIPS need to track at an instance becomes tremendous. For example, hospitals want to track all the important medical devices so that they can prevent loss of these devices as well as manage the number of devices that are currently in use or stored. In addition, at many enterprises, during peak hours, the number of mobile devices can increase drastically. However, tracking many devices simultaneously is challenging due to the design of the IIPS in which all location computations are performed at a centralized server having limited computational resources. Therefore, scaling the IIPS efficiently with a large number of tracked devices is critical towards enabling location-based services at large enterprises.

How accurately the IIPS can localize and track devices determines the type of location-based services that can be enabled. In particular, asset tracking that allows a building operator to track a valuable asset such as a small medical device requires sub-meter location accuracy. Head count allows a building operator to allocate resources efficiently based on the number of users in an area. This service often requires room-level accuracy. A navigation service that provides a user directions to

a destination requires IIPS to accurately determine his or her path as well as location. However, due to noisy, temporally sparse Wi-Fi measurements that the IIPS obtain to track a device, it is extremely challenging to determine the exact device location, the room wherein the device locates, or its current path. Therefore, improving the localization and tracking accuracy of the IIPS is important to enable various location-based services.

In summary, there is a strong need for accurate and scalable localization and tracking from the IIPS in order to enable various location-based services in large indoor environments. In this dissertation, the research problem is how to achieve accurate localization and tracking of many devices simultaneously. We next describe the main challenges that must be addressed to solve the problem.

1.3 Research Challenges

Solving this problem is difficult due to three main challenges. First, the IIPS need to track many devices with limited computational resources. Second, the IIPS need to localize and track devices accurately given that Wi-Fi measurements are often noisy. Third, temporally sparse, non-periodic measurements have weak correlations between consecutive measurements, which in turn makes it difficult to leverage these measurements to improve localization and tracking accuracy. Below, we describe these challenges in detail.

1.3.1 Limited Computational Resources

The IIPS need to track a large number of devices within enterprises simultaneously with limited computational resources. Due to data privacy concerns, some enterprises want the IIPS to be deployed on-site, i.e., locally at the enterprises. Therefore, the computational resources of the IIPS are strictly constrained. Provisioning computa-

tional resources at deployment time is difficult. It requires a careful estimation of how many devices the IIPS must support over a long-term period. Also, during peak hours at an enterprise, the number of devices can increase many folds.

Though the IIPS can be deployed on-cloud with computational resources allocated dynamically, there are cost tradeoffs. Merely allocating more resources to concurrently track devices can increase the IIPS cost significantly, which limits the widespread adoption of the IIPS technology. The estimated cost for operating a minimal cluster, databases, storage, and bandwidth utilization on Amazon Web Service is high, about $8,000 per month for tracking ten thousand devices every four seconds on average [22]. Therefore, whether the IIPS are deployed on-site or on-cloud, the IIPS can suffer from limited computational resources, which makes it challenging to scale the IIPS cost-effectively with the number of tracked devices.

1.3.2 Noisy Measurements

One of the well-known challenges in providing accurate localization and tracking is noisy measurements. The measured Wi-Fi signals at APs are often noisy due to unpredictable multipath propagation and attenuation of the signals in indoor environments [33,73,78]. As a result, the location estimates from the signals often scatter around the corresponding actual locations.

1.3.3 Temporally Sparse, Non-Periodic Measurements

The third challenge is in aggregating temporally sparse, non-periodic Wi-Fi measurements to improve the localization and tracking accuracy. Often, multiple noisy measurements can be easily aggregated to improve localization and tracking accuracy if these measurements are sampled at a high sampling rate, such as multiple samples per a few seconds [65, 74]. However, as the IIPS obtain measurements in a best

effort manner (as described in Section 2.3), the measurements are often temporally sparse and non-periodic. As a result, there is a weak correlation between these measurements. For a moving device, it is challenging to combine historical and current measurements to improve localization and tracking accuracy.

1.4 Limitations of Existing Solutions

Although there have been attempts to address the above challenges, they mostly address each challenge in isolation, focusing on either the limited computational resources [1, 51, 56, 79] or the noisy measurements [33, 73, 78].

• *Prior work on limited computational resources.* Several approaches have been developed to improve the scalability of the IIPS with respect to their constrained computational resources. Basically, the required amount of computational resource is proportional to (i) the number of tracked devices, (ii) the frequency of location computations per device, and (iii) the computational cost per location computation. Simply reducing the location computation frequency can have a significant impact on tracking highly mobile devices whose locations change frequently. Another approach is to reduce the computational cost per location computation with negligible impact on location accuracy. In particular, for Angle-of-Arrivals (AoA) based localization, the cost of computing accurate AoA from the latest Wi-Fi measurements such as channel state information (CSI) and phase vector is expensive. Prior work [1, 56] proposed solutions to achieve AoA with similar accuracy but lower computational cost. Moreover, AoA-based localization and other localization methods often have high location computation cost. This is because they solve optimization problems to search for the most likely location corresponding to the latest measurements. Prior work [51, 79] proposed solutions to reduce the search space without affecting the location accuracy.

• *Prior work on noisy measurements.* Most prior work has focused on improving location accuracy based on the latest explicit Wi-Fi measurements. Particularly, prior work on AoA-based localization [33, 78] concentrated on computing AoA from the current measurements (such as CSI) such that the computed AoA are less variant to multipath propagation of the signals. In addition, most state-of-the-art localization approaches [33,73,78] often addressed the noisy measurements by solving optimization problems to find the most likely location with respect to the measurements.

• *Summary.* We emphasize that most prior works have focused on either improving *scalability* of the IIPS or improving their localization and tracking *accuracy*. They have addressed either the limited computational resources of the IIPS or noisy measurements. Their solutions narrowed down to improving a limited subset of localization algorithms that use only the latest explicit Wi-Fi measurements at multiple APs. In this dissertation, we make the case for developing solutions that improve a richer set of localization algorithms using a wider variety of information. We propose a novel approach that improves both scalability and tracking accuracy by addressing all of these three challenges simultaneously. We give an overview of our approach in the next section.

1.5 Our Solution: Context-Aware IIPS

Instead of relying only on explicit Wi-Fi measurements at the IIPS, we investigate what other useful information at the IIPS can be exploited to address the problem. We focus on investigating information related to a device's location at a time, named *device context*. The IIPS can adapt to the context to localize and track the device efficiently and accurately. Figure 1.2 shows several device contexts. In our work, we focus on exploiting two important device contexts that can be derived from data available at the IIPS.

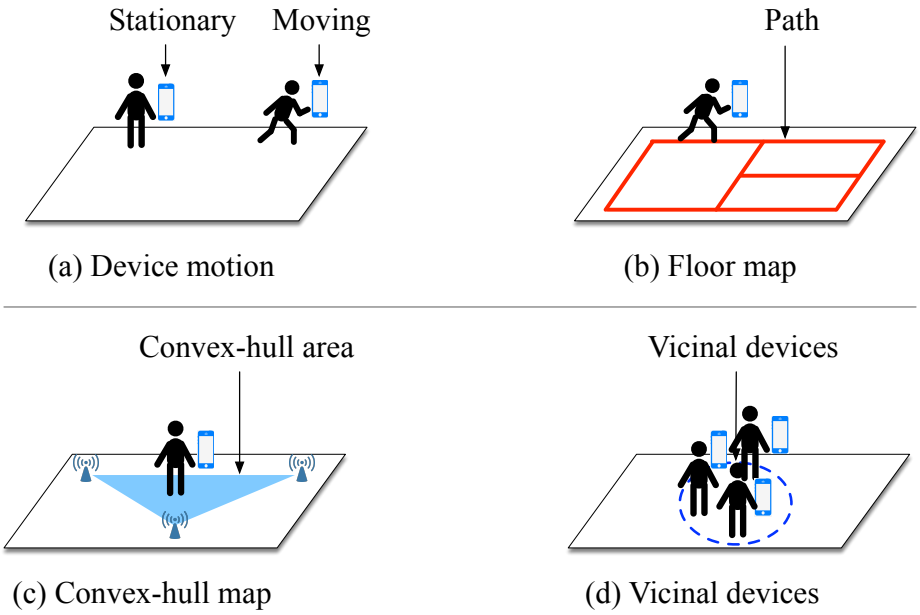


Figure 1.2: At a time, at a location, a device can have several contexts: (a) device motion indicates whether the device is stationary or moving (b) floor map indicates possible paths which the device user can be on (c) convex-hull map indicates whether the device is inside a convex-hull area formed by access points that receive signals emitted from the device [49] (d) vicinal devices indicates other devices in the vicinity of the device.

- *Device motion context.* We observe that devices can significantly change their motion over time. Particularly, devices are often stationary in some periods. It is needless to recompute location of these devices repeatedly. Therefore, IIPS can prioritize computing locations for moving devices given the IIPS' limited computational resources. IIPS can also aggregate several temporally sparse Wi-Fi measurements for improving localization accuracy. Therefore, the device motion context is very useful to address the scalable and accurate localization and tracking with the IIPS. However, it is challenging to derive this hidden context from only Wi-Fi measurements available at the infrastructure.

• *Floor map context.* A digital floor map, which is available at the infrastructure, represents paths or obstacles. For moving devices, device users often follow some paths and can not cross obstacles such as walls. We can constrain noisy, temporally sparse location estimates on the paths. Therefore, the floor map is an important context for improving tracking accuracy. However, in an indoor environment, often there are many possible paths that the users can take. Therefore, given noisy and temporally sparse location estimates, it is challenging to leverage this context to improve tracking accuracy.

In this dissertation, we present the *first* solution that extracts and leverages the device contexts to address all three challenges in conjunction. Figure 1.3 shows the summary of our solution and comparison with existing solutions. *The thesis of this dissertation is that embedding context-aware capabilities in the IIPS enhances its performance in tracking many devices simultaneously and accurately.*

1.6 Contributions

Our main contributions that help realize the significant impact of leveraging device contexts are:

- **Infrastructure-based motion detection:** We design, implement and evaluate MotionScanner, which includes novel feature-based and end-to-end deep learning motion detection models to detect device motion solely from noisy, temporally sparse, and partial Wi-Fi measurements at access points. The key idea is to exploit temporal patterns of measurements of different Wi-Fi attributes (signal strengths and phase vectors) across multiple APs to detect a device's motion accurately in real time. We further integrate MotionScanner into the IIPS so that the IIPS can exploit previously computed locations effectively to improve localization accuracy, and skip unnecessary location computations. Finally, we

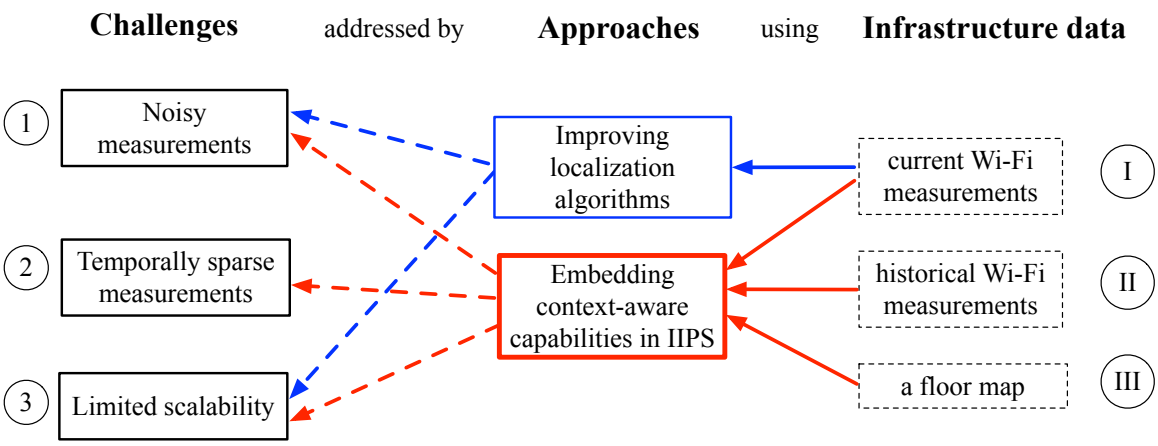


Figure 1.3: Summary of the solutions. Existing approaches that improve localization algorithms use data (I) to address challenges (1) and (3). Our work embeds context-aware capabilities (e.g., device motion detection and map matching) in IIPS by using data (I), (II), and (III) to address all three challenges simultaneously. Specifically, device motion detection uses data (I) and (II) to determine device motion to address challenges (1), (2), and (3). Map matching uses data (I), (II), and (III) to address challenges (1) and (2).

evaluate MotionScanner with dataset collected from real-world deployments of the IIPS at two enterprises, and show that MotionScanner achieves 83% motion detection accuracy while saving 80% of computational resources. To the best of our knowledge, this is the first system to use device motion detection to significantly improve the IIPS. It is also the first system to apply recent advances in deep learning for device motion detection.

- **Infrastructure-based map matching:** We design, implement and evaluate the first infrastructure-based map matching approaches to select a correct predefined path segment to align the current location estimate. We propose two different approaches: a *graph-based approach* and an *image-based approach*.

- In the first approach, we represent a floor map as an undirected graph. Then, we develop a Hidden Markov model that applies a combination of several constraints including a point-to-curve constraint, a travel distance constraint, and a number of turns constraint to select the correct path segment. The last constraint is based on our preliminary human mobility study.
- In the second approach, we represent the floor map as a binary image whose pixels inside the path segments have value 1. We also represent a location estimate and its associated distribution of estimation error on an image, called a location image. Given this input representation, we develop a 3D convolutional encoder-decoder neural network to address the problem.

To evaluate these approaches, we use two different metrics: location accuracy and our proposed path-alignment accuracy, which is based on how close the aligned current location estimate to the corresponding actual location. We evaluate the first approach by using datasets collected at a lab and a workplace, and the second approach by using our simulated datasets. Our results show that our graph-based approach can improve path-alignment accuracy, compared to prior graph-based approaches. About 40% of location estimates are now aligned on the path taken by a user and 4 meters behind the corresponding user locations. Compared to the graph-based approach, the image-based approach consistently achieves much better location accuracy. However, path-alignment accuracy varies relative to backward path length of the path-alignment accuracy metric. To the best of our knowledge, this is the first system to explore the use of different representations of the floor map to enable infrastructure-based map matching.

1.7 Dissertation Overview

In the following chapters, we provide the background useful to understand our work (Chapter 2), the infrastructure-based motion detection model (Chapter 3), and the infrastructure-based map matching algorithm (Chapter 4). In Chapter 5, we summarize the dissertation and describe directions for future work.

The content of the dissertation is based on two major conference publications [67, 69] and one poster [70]. We have developed an interesting location-based service that the IIPS can provide client count prediction that is not included in this dissertation [68].

2 Background

This chapter describes the network architecture of typical enterprise IIPS [16]. We also describe Wi-Fi attributes of interest and how they are collected by the IIPS. Knowing how the Wi-Fi attributes are measured is key to understanding why they are noisy and temporally sparse. In addition, we provide an overview of infrastructure-based localization methods that the IIPS can use to localize and track Wi-Fi devices.

2.1 IIPS Architecture

Most enterprise buildings have Wireless Local Area Network (WLAN) infrastructure wherein Wi-Fi devices can communicate with each other and a server through wireless connection. A typical enterprise WLAN infrastructure consists of access points (APs) and WLAN controllers [14,19]. APs are devices that communicate with Wi-Fi devices (phones, Wi-Fi tags) to allow sending and receiving data from a wired network. Each AP is connected to one or multiple controllers. Each controller is responsible for managing a set of APs to improve WLAN's performance such as Wi-Fi coverage and security [19].

Figure 2.1 illustrates an architecture of the IIPS on top of an enterprise WLAN infrastructure [17]. Each AP measures attributes of Wi-Fi signals emitted from Wi-Fi devices and forwards the measurements to a WLAN controller. The controller aggregates and then forwards the measurements received from multiple APs to a

centralized location server (LS) deployed on-site or on-cloud. Based on the measurements, the LS localizes and tracks Wi-Fi devices. The estimate locations of these devices can then be forwarded to another sever that performs location analytics and location-based services.

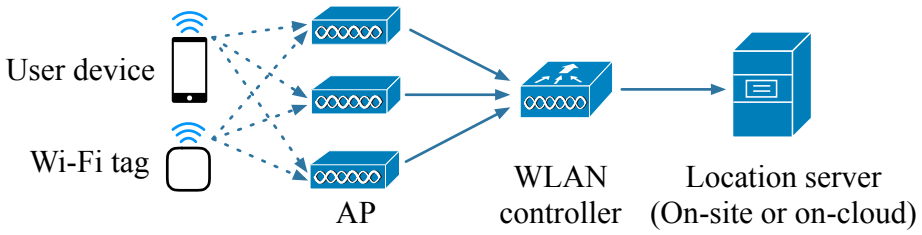


Figure 2.1: Network architecture of the IIPS on top of an enterprise WLAN infrastructure

2.2 Wi-Fi Attributes

In this section, we describe the attributes of Wi-Fi signals that a commercial-off-the-shelf (COTS) AP [15, 23, 24] can measure for performing device localization. Though different AP models can have different antenna designs, they often have two groups of antennas: serving antennas for transmitting or receiving signals from Wi-Fi devices and circular-array antennas. Figure 2.2 shows that the COST AP [23] has 4 serving antennas and 32 quasi-circular-array antennas. These antennas can measure two main Wi-Fi attributes: received signal strength indicator (RSSI) and angle-of-arrivals (AoA) phase vector consisting of phase values.

- RSSI indicates the power of a Wi-Fi signal when it arrives at a serving antenna of the AP [4].
- AoA phase vector indicates the phases of a Wi-Fi signal when it arrives at different circular-array antennas of the AP [27, 78]. The phase vector is computed

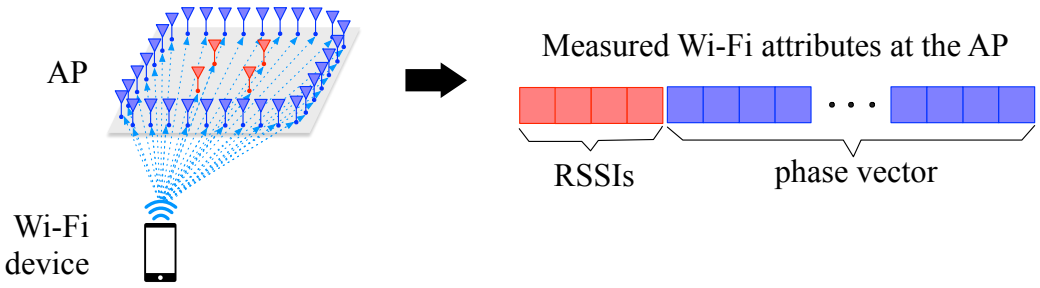


Figure 2.2: Measured Wi-Fi attributes at a COTS AP

from channel state information (CSI) [10] measured at the physical layer of the AP [27, 78].

2.3 Measurement Process

In this section, we describe how the Wi-Fi attributes are measured by the IIPS, which is necessary in order to understand why the Wi-Fi measurements are temporally sparse and non-periodic. The IIPS sample Wi-Fi measurements in a best effort manner without requiring any specific applications installed on a tracked device.

- For RSSI measurements, a group of APs in the vicinity of a device measures the RSSI of the signal emitted from the device whenever the device transmits a packet. Such packet can be a probe request for APs [45], an uplink data packet, or a response to a request from the AP when the device's radio is active [15, 18].

- For AoA phase vector measurements, a group of APs measures AoA of signals emitted from a device when a master AP exchanges packets with the device. The master AP selects each of its associated devices sequentially, and each AP in the group is also selected as the master AP in a round robin way [21].

However, for both RSSI and AoA phase vector measurements, because of unpredictable interference and the CSMA nature of Wi-Fi networks as well as the uncontrolled device behavior, the IIPS can not guarantee that any measurements are

conducted at scheduled time instants. Neither can it guarantee that all packet exchanges (e.g., between a device and a master AP) are successful. As a result, we will observe uncertain delay between consecutive Wi-Fi measurements of the same device which we called *temporally sparse and non-periodic measurements*, as well as partial measurements which we also name *missing measurements*.

2.4 Infrastructure-based Localization

Various approaches have been proposed for infrastructure-based localization. In this section, we describe the three most popular localization approaches that the IIPS can perform: RSSI-based localization, fingerprinting-based localization, and AoA-based localization. These terminologies have been proposed in [33]. This overview is necessary to understand why accurate localization methods can be computationally expensive.

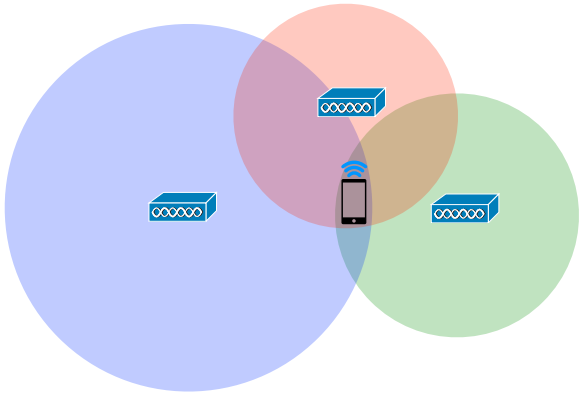


Figure 2.3: RSSI-based localization: A device location is estimated by performing trilateration using the estimated distance between each AP and the device.

RSSI-based localization: From RSSI of a Wi-Fi signal measured at an AP, a distance between the device that transmits the signal and the AP can be estimated by using a path-loss model which models how the transmission power of the signal

degrades over the propagation distance. Given the estimated distances from the device to at least three APs, a trilateration method can be used to estimate the device location (Figure 2.3). However, RSSI measurements are often noisy due to mutlipath propagation of the signal and interference of Wi-Fi signals at the APs' antennas. The median location accuracy of RSSI-based approaches ranges from 2 m to 4 m [4, 13]). The main advantage of these approaches is that they have low computational cost.

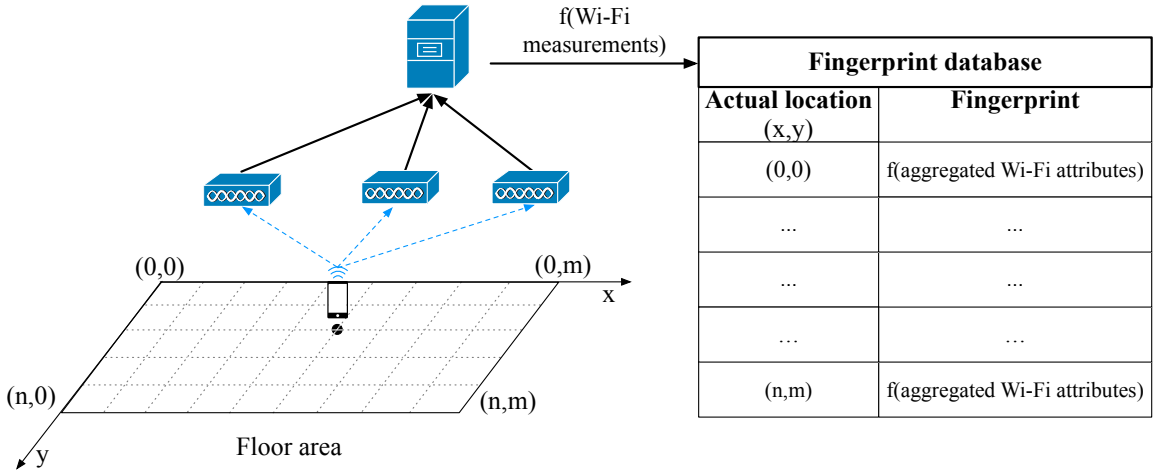


Figure 2.4: Fingerprinting-based localization: A device location is determined by matching the pattern (a function) of the measured Wi-Fi attributes to fingerprints stored in a database.

Fingerprinting-based localization: In the calibration step, a floor area is divided into small grids (such as 1 m x 1 m grids), a device is placed at each grid's center (or grid point) for a period that can last several seconds or minutes. During this period, the Wi-Fi attributes measured at multiple APs in a vicinity with the device are recorded. A unique pattern (fingerprint) associated with the location is computed and recorded. As a result, the calibration step builds a fingerprint database consisting of the mappings between fingerprints and grid points. In the matching

step, localization is performed by matching the pattern of Wi-Fi measurements to the fingerprint database (Figure 2.4). The disadvantage of this approach is that the calibration step is repeated when the indoor environment changes. The location accuracy of fingerprinting-based approaches depends mainly on the grid's granularity, how recent the calibration step is, what Wi-Fi attributes used to compute a fingerprint, and a density of APs. Median location accuracy of state-of-the-art approaches ranges from 1 m to 2 m (with a grid size of about 1 m x 1 m) [73].

Compared to RSSI-based localization approaches, fingerprinting-based approaches often achieve better localization accuracy. However, these approaches also have higher computational cost, which is mainly because the matching step needs to match the current Wi-Fi measurements with many fingerprints in the database [79].

AoA-based localization: An angle-of-arrival is defined as the angle between the propagation direction of an incident signal and some reference direction (orientation) pre-defined at an AP. A simple approach to obtain the angle is to compute the difference between the phases of the signal measured at the array antennas of the AP (as described in Section 2.3.1 by Jie et al. [78]). With reported AoA measurements from at least two APs for the same target device, a triangulation method is applied to calculate the device's location (Figure 2.5). However, due to multipath propagation of signals, this simple approach does not compute the AoA precisely. State-of-the-art approaches [33, 78] achieve high resolution AoA by essentially applying algorithms such as MUSIC and ESPRIT [56, 57]. Given the estimated AoA from multiple APs, these approaches find the location that maximizes the likelihood of observing these estimates. The best median location accuracy of state-of-the-art approaches ranges from 0.4 m to 1.6 m.

Though AoA-based approaches can achieve high localization accuracy, their computational costs are often much greater than RSSI-based approaches because of two

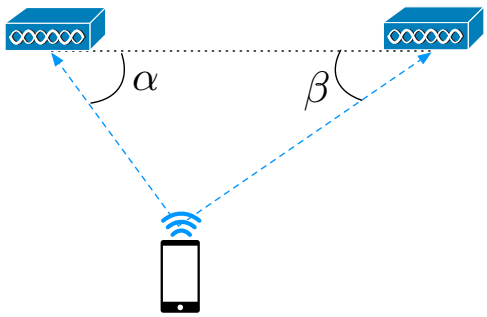


Figure 2.5: AoA-based localization: A device location is estimated by performing triangulation using the estimated AoA at two APs.

main reasons. First, the complexities of the algorithms for estimating an angle-of-arrival increase polynomially with the number of array-antennas of an AP [1]. To achieve high resolution AoA, large number of (physical or virtual) antennas are often required [33,57,78]. Second, finding the device location that maximizes the likelihood of observing the estimated AoA requires solving an optimization problem.

In our work, we use a combination of RSSI-based trilateration and phase-based AoA method to localize and track the devices [27,51,78]. The method achieves median localization accuracy ranging from 1 m to 3 m for real-world deployments at many enterprise buildings (including retails, airports, and workplaces) having about one AP per 15 m x 15 m (which is required to guarantee good Wi-Fi coverage [20]). The method also achieves sub-meter accuracy for areas having line-of-sight between the device and several APs.

3 Infrastructure-based Motion Detection

This chapter describes our first contribution that is the infrastructure-based motion detection to determine a device's motion as moving or stationary. We design, implement and evaluate MotionScanner, which includes novel feature-based and end-to-end deep learning motion detection models to detect device motion accurately by using only noisy, temporally sparse, and partial Wi-Fi measurements at access points. The key idea is to exploit the relationship between temporal patterns of measurements of different Wi-Fi attributes (signal strengths and phase vectors) across multiple APs and the device motion. We also show that MotionScanner can enable skipping unnecessary location computations and improve localization accuracy for stationary devices.

3.1 Motivation

In this section, we first describe our insights on why exploiting device motion context is important to address the problem of scalable and accurate localization and tracking.

• *Scalability:* The IIPS need to track a large number of devices in real time without reducing location accuracy and increasing the cost significantly. For large enterprise buildings, the number of devices can also increase many folds during peak hours. Though extra computational resources can be allocated dynamically by using a cloud computing service, as we deployed the location server on Amazon Web Service,

the cost of running a minimal cluster, databases, storage, and bandwidth is high (about $8,000 per month for tracking ten thousand devices every four seconds on average). We observe that within many enterprise buildings, people and devices are often stationary for long durations. For example, in office spaces, people are typically stationary 75% of the time [37]. During peak hours at airports and retails, people are often stationary at waiting lounges or waiting queues. Moreover, Wi-Fi tags are often stationary for long periods of time. By exploiting the device motion, the IIPS can scale cost-effectively by not computing locations of stationary devices repeatedly.

• *Location accuracy:* The IIPS need to localize and track a device accurately. Most state-of-the-art localization approaches have focused on improving location accuracy by using only the latest Wi-Fi measurements [33, 78]. Besides, several filtering methods [65, 74] can be applied to exploit historical measurements to improve the current location estimate. However, as measurements obtained by the IIPS are temporally sparse and non-periodic, there are weak correlations between consecutive location estimates when the device is moving. Therefore, it is essential to determine the device's motion. If the device is stationary, the filtering methods can be applied directly to improve the current location estimate.

This work addresses the problem of detecting device motion at an infrastructure side by using only Wi-Fi measurements that the IIPS receive. Exploiting device sensor data [28, 48, 80] can further improve motion detection accuracy. However, this approach does not fit well with the IIPS design as it requires (i) all tracked devices to run motion detection applications (ii) network protocols to support collecting device data (iii) explicit permission of device users for storing their data (iv) correctness and trustworthiness of motion data sent from devices. This work focuses on investigating to what extent detecting device motion by using only the Wi-Fi measurements can improve the IIPS performance.

3.2 Contributions

We present **MotionScanner** that enables the motion-aware IIPS by using only Wi-Fi measurements (RSSIs and phase vectors) that the IIPS receive, and addresses the three main challenges: temporally sparse and non-periodic measurements, noisy measurements, and missing measurements (Section 3.3). Our main contributions can be summarized as below.

- We developed feature-based and deep learning-based models that exploit temporal patterns of measurements from multiple APs to detect device motion accurately in real time (Section 3.6). We focus on the generalizability and simplicity of our models. Our models consist of three main steps: feature extraction for extracting temporal patterns of each AP's measurements, feature aggregation for aggregating temporal features across multiple APs, and modeling for learning the relationship between the features and device motion. The main novelty of our models is the method of computing and aggregating the features, especially phase correlations, effectively regarding temporally sparse, multipath, and missing measurements.

- We evaluated our models by using dataset collected from real-world deployments of the IIPS at two different enterprise settings: a retail and a cafeteria (Section 4.8). Our results showed that:

  - Our feature-based models achieve 83% motion-detection accuracy on average by using both features extracted from RSSIs and phase vectors. We also showed that RSSI features proposed in prior work [32, 34, 43] achieve at most 75% accuracy on average with our dataset. Moreover, our feature-based models trained by using data collected in one building can be applied

to another building with negligible accuracy reduction.

  - Our end-to-end deep learning (E2E) models can exploit temporal patterns of RSSIs and phase vectors effectively to detect device motion. By using only RSSIs, our E2E model can further improve motion detection accuracy by 3%, compared to a feature-based Random Forest model. By using only phase vectors, our E2E model can further improves the accuracy by 5%, compared to using the phase correlation. By using both RSSIs and phase vectors, our E2E model achieves comparable accuracy to the feature-based model.

- We showed that MotionScanner can improve the IIPS performance in terms of location accuracy and scalability. Particularly, MotionScanner reduces the number of location computations by as much as 80% without any impact on location accuracy.

3.3 Challenges

There are three main challenges that we address in detecting device motion by using Wi-Fi measurements reported from the infrastructure side.

• *Temporally sparse and non-periodic measurements*: As discussed in Section 2, the IIPS sample Wi-Fi measurements in the best effort manner. Therefore, the measurements are temporally sparse and non-periodic. In our dataset (Figure 3.8d), the average sampling rate of RSSI measurements is about one sample per 5 seconds with a standard deviation (SD) of 4 seconds. The average sampling period of phase measurements is about one sample per 6 seconds with an SD of 6 seconds. Figure 3.2 illustrates how measurements reported by APs in the vicinity of a device over time and the histogram of the measurements. Given the temporally sparse and non-periodic

measurements reported by APs, it is challenging to detect device motion at a time step, especially when the device stays or moves in short time intervals.

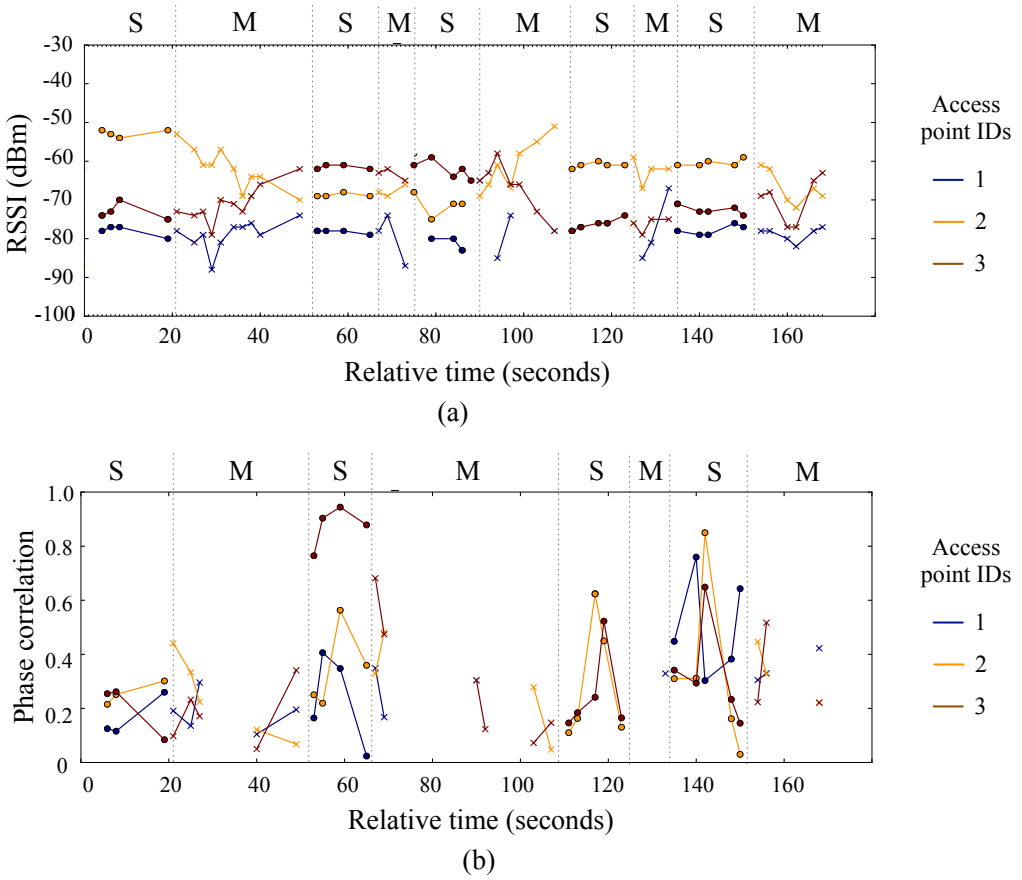


Figure 3.1: (a) Measured RSSIs and (b) computed phase correlations at three APs deployed in a cafeteria (shown in Figure 3.8a) when a phone is carried around by a user alternating between stationary (S) and moving (M).

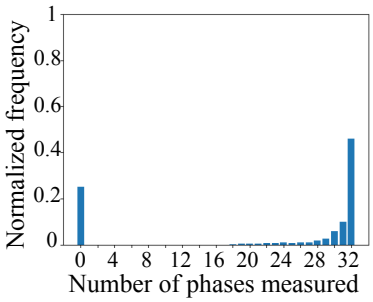
- *Noisy measurements*: In an indoor environment, Wi-Fi measurements at an AP are often noisy due to unpredictable signal attenuation and multipath signal propagation. For RSSI measurements, Figure 3.1a shows RSSI measurements vary when a phone is moving. However, there are episodes in which the device is stationary but RSSI measurements fluctuate at a similar level compared to when the device is moving. For phase measurements, Figure 3.1b shows the correlations of consecutive

phase vectors. When the device is stationary, the correlations are relatively higher. However, they fluctuate significantly. Therefore, it is challenging to detect the device's motion accurately given the noisy measurements.

Time	1	6	12	15	20	25	27	31	38	40	…
RSSIs	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	…
Phase vectors	✓	×	✓	×	✓	✓	✓	×	✓	✓	…

Measurements reported by APs over time (x represents no measurements)

(a)



(b)

Figure 3.2: (a) Measurements reported by APs in the vicinity of a device over time. (b) Histogram of the average number of measured phase values per AP. About 25% of the time, APs report only RSSIs. About 75% of the time, the APs report both RSSIs and phase vectors. Within that, about 50% of the phase vectors have full 32 phase values measured at the 32 circular-array antennas of each AP.

- *Missing measurements*: We observe three main cases in which measurements at an AP are missing. First, when a device stays far from the AP or is moving, we often observe intermittent measurements at the AP over time [32]. Figure 3.1 represents missing measurements over time as white gaps between markers in each motion (stationary or moving) episode. Second, the device may not respond to all request packets from the AP during the phase measurement process (as described in Section 2). Figure 3.2b shows that about 25% of phase vectors have missing phase values. Third, the device can switch between different frequency bands (2.4GHz and 5.0GHz) when sending Wi-Fi signals. The switching frequency is device dependent.

3.4 Related Work

We first describe motion types, then categorize motion detection literature based on motion types, and finally highlight where MotionScanner fits in the overall literature as shown in Figure 3.3. According to Sun et al. [61], there are three types of object motion: stationary, micro motion, and macro motion. Stationary means an object is completely static. Micro motion means the object only moves a small distance (less than one meter) or some part of the object (human hand) moves. Macro motion means the object moves more than one meter. Given these object motion definitions, there are two main categories of motion detection problems that prior work has focused on: (i) stationary (including micro motion) versus macro motion, and (ii) stationary versus motion (including micro and macro). Our work MotionScanner belongs to the first category.

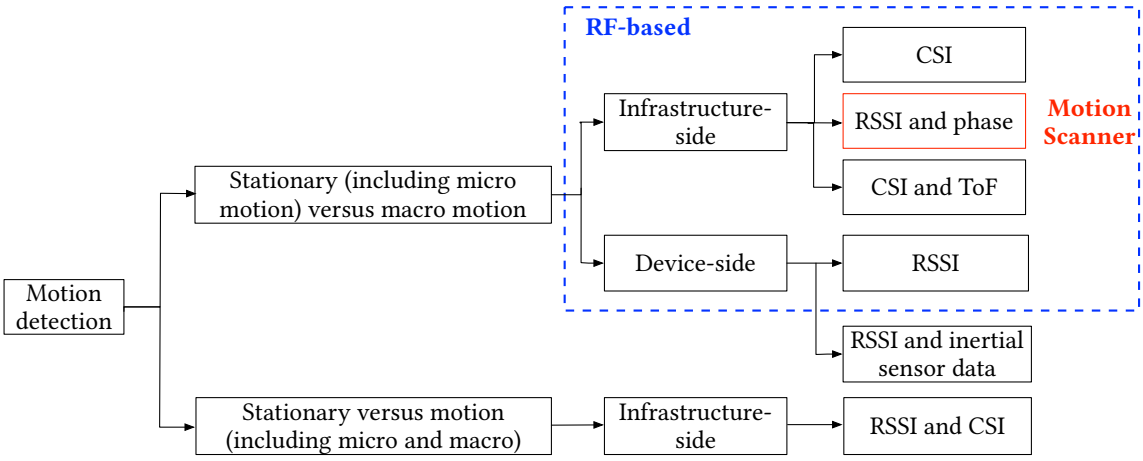


Figure 3.3: Related work on motion detection

Stationary (including micro motion) versus macro motion: In this category, motion detection is performed either at an infrastructure side or at a device side (Figure 3.3). MotionScanner is performed at an infrastructure side by using RF data that the IIPS measure in the best effort manner (Section 2). Therefore, the

IIPS sample Wi-Fi measurements at a non-periodic and low rate (about 0.2 Hz for RSSIs and 0.15 Hz for phase vectors). This is one of the unique challenges in detecting device motion with the IIPS measurements (Section 3.3). We summarize motion detection approaches in this category in Table 3.1 as well as describe them below.

		Infrastructure	Measurement	Sampling rate	Real-time	Reported accuracy	Use cases
Infrastructure side	**Motion Scanner**	1) Multiple APs WLAN 2) AP density: ~ 1 AP per 15m x 15m 3) Real-world deployments	RSSI and phase	Low - RSSI: ~0.21 Hz - Phase: ~0.15 Hz	Yes	83%, with 1) Low, non-periodic sampling 2) Missing data	- Scalability - Localization
	Sun et al. 2014	1) Single AP WLAN 3) Test bed (office)	CSI and ToF	High (~5 Hz)	No (4 second delay)	98%, with 1) High, periodic sampling 2) No missing data	- Client roaming - Rate control - MIMO beamforming
	Wu et al. 2015	1) 1 transmitter and 1 receiver 3) Test bed (office)	CSI	High (50 Hz)	Not available	95%, with 1) High, periodic sampling 2) No missing data 3) Single tracked object	Human monitoring
Device side	Muthukrishnan et al. 2009	1) Multiple APs WLAN 2) AP density: not available 3) Test bed (indoor and outdoor)	RSSI	Medium (0.4 Hz)	Not available	92%, with 1) Medium, periodic sampling 2) No missing data	Localization
	Krumm et al. 2004	1) Multiple APs WLAN 2) AP density: not available 3) Test bed (house)	RSSI	High (3 Hz)	Yes	87%, with 1) High, periodic sampling 2) No missing data 3) Human mobility model	Localization

Table 3.1: Prior work on using RF signals to classify stationary (including micro motion) versus macro motion.

Prior infrastructure-side approaches [61, 76] detect device motion by requiring tracked devices or transmitters to send RF signal periodically at 5Hz and 50Hz, respectively. Moreover, [61] needs to aggregate many measurements in 4 seconds to detect device motion with 98% accuracy. [76] achieves 95% accuracy but assuming

there is only one tracked device in an environment. Both of these approaches detect device motion by using measurements (channel state information or time of flight) reported from a single receiver deployed in test beds (office spaces). MotionScanner combines different temporal correlations of measurements (RSSIs and phase vectors) reported from multiple commercial-off-the-shelf (COTS) APs to detect device motion in real time.

Prior device-side approaches detect device motion by using data (RSSIs or inertial sensor data) collected from applications running on a device. Prior work [32,34] samples RSSI measurements periodically at 0.4 Hz and 3 Hz, respectively. The prior work proposed several RSSI features in a frequency domain and a time domain for detecting device motion. Extracting the features in the frequency domain requires measurements sampled periodically, while measurements from the IIPS are non-periodic. For the features in the time domain, we exploit these features and show that using these features achieves much lower accuracy in our dataset (Section 4.8). Prior work [34] also assumes a fixed frequency of a person's movement in a test bed. Without this assumption together with a low and non-periodic sampling rate, MotionScanner exploits not only a combination of RSSI features but also a phase feature to detect device motion accurately in real-world deployments of the IIPS. Moreover, as we discussed in Section 4.1, requiring applications running on every tracked device does not fit well with the IIPS design.

Stationary versus motion (including micro and macro): Recent non-invasive (device-free) solutions have focused on detecting either micro motion or macro motion of tracked objects for healthcare applications such as in-home elderly or child monitoring and gesture recognition [30, 31, 72, 77]. Custom hardware [30, 31] or pairs of a transmitter and receiver [72,77] are required to analyze RF measurement changes caused by motion. These solutions are limited to small, static environments having

a single or very few tracked objects (less than 6) [31]. Also, they do not distinguish between micro motion and macro motion.

3.5 Problem

We first formalize the motion detection problem. Table 3.2 introduces the notation we use to define the motion detection problem. Figure 4.1 depicts the problem.

Symbol	**Description**
time_t	Timestamp of time step t
RSSIA_t^n	Signal strength measured in 5GHz at an AP n at time_t
RSSIB_t^n	Signal strength measured in 2.4GHz at an AP n at time_t
RSSIs_t^n	$[\text{RSSIA}_t^n, \text{RSSIB}_t^n]$
$\text{phase}_t^{n,p}$	Phase value measured at antenna p at AP n at time_t
phases_t^n	Phase vector measured at an AP n having P antennas $[phase_t^{n,1}, ..., phase_t^{n,P}]$
a_t^n	Data measured at an AP n at time_t $[\text{RSSIs}_t^n, \text{phases}_t^n]$
a_t	Data measured at N APs deployed on a floor at time_t $[a_t^1, a_t^2, ..., a_t^N]$
m_t	Device motion at time step t $m_t = 0$: the device is stationary (negative) $m_t = 1$: the device is moving (positive)

Table 3.2: Notations

Online device motion detection. Given the time series $a_1, ..., a_t$ of Wi-Fi measurements per device reported from a set of access points within a floor until the current time t, the IIPS needs to classify the device motion m_t at the time t as stationary (including micro motion) or moving i.e., macro motion. The definitions

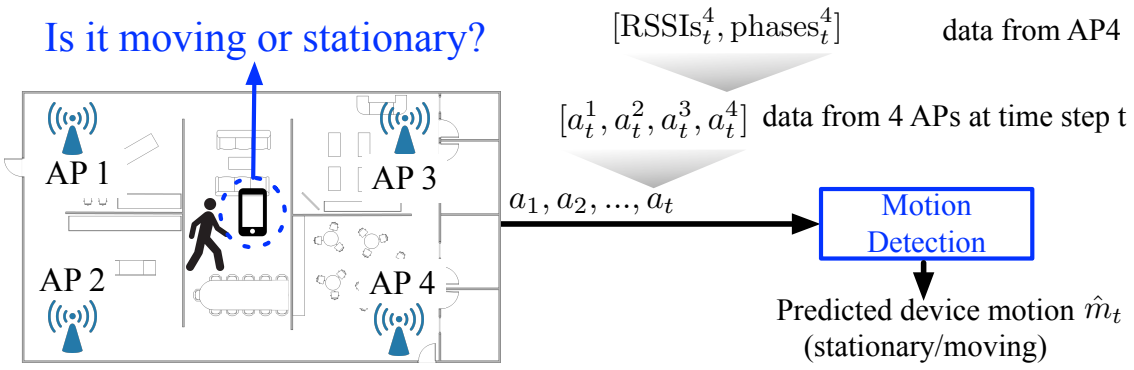


Figure 3.4: Motion detection problem

of motion types are in the first paragraph of Section 3.4. In this paper, we interchangeably use motion detection, motion prediction or motion classification to mean the same thing.

3.6 Approach

To address the online motion detection, we propose two approaches: feature-based approach and deep learning-based approach.

3.6.1 Feature-based Approach

The feature-based approach consists of three steps: feature extraction, feature aggregation, and modeling. The feature extraction step extracts temporal features from the measurements reported from each AP until time_t such that each feature correlates with device motion at time_t. To reduce the effect of noisy measurements and missing measurements on the correlation, different temporal features extracted from measurements at multiple APs' are aggregated to generate robust spatial-temporal features. Finally, the modeling step trains models to learn the relationship between the features and the device motion. Below, we describe these steps in detail.

3.6.1.1 Feature Extraction

At the current time step t, the feature extraction step extracts temporal features from the Wi-Fi measurements reported from each AP in a time window from $t - w$ until t. The window size parameter w is derived from a training dataset to achieve optimal classification accuracy.

• *RSSI features:* We experimented with a variety of RSSI features. In Table 3.3, we list the RSSI features that contribute to improving the motion detection accuracy together with our key observations for each feature. Other tested features in time domain include Tanimoto distance, Spearman's correlation, and Euclidean distance over RSSIs. However, those features do not contribute significantly to improving the accuracy of motion detection.

RSSI features	**Description**
Visibility of AP	The ratio between the number of measured RSSIs and the window size w [32]. We observe that the AP visibility reduces when a device is moving.
Consecutive RSSIs	The RSSIs at time steps $t-1$ and t, which will be used to compute the correlation of RSSIs across APs in the feature aggregation step.
Consecutive RSSI difference	The absolute difference between two consecutive RSSI measurements at times $t-1$ and t. If an access point does not report RSSI at either of the times, it will not be used in the feature aggregation step.
SD of RSSIs	The standard deviation of RSSIs measured within the window w if the RSSI at t is available and the number of available RSSIs is greater than 3.
SD of RSSI differences	By first computing the absolute differences between the single current RSSI with all previous RSSIs in the window w and then computing the standard deviation of those differences.

Table 3.3: RSSI features

• *Phase correlation feature:* Phase correlation represents the similarity between two phase vectors measured by an AP n at two different time steps. If there is

a strong correlation between the phase vectors at the current time step t and the previous time step $t-1$, a device is likely to be stationary. However, as mentioned in Section 3.3, we often have missing phase vectors due to the lower sampling of the phase vectors compared to RSSIs. Therefore, instead of considering only the phase vector at $t-1$, we consider the phase vector at time step t' in a window of w measurements ($t' \in [t-w, t-1]$). We define the phase correlation below.

$$\text{corr}(\text{phases}_t^n, \text{phases}_{t'}^n) = \frac{|\sum_p e^{2\pi j(\text{phases}_t^{n,p} - \text{ phases}_{t'}^{n,p})}|}{c} \tag{3.1}$$

The phase correlation is computed by using only the pairs of phase values denoted as $\text{phases}_t^{n,p}$ and $\text{phases}_{t'}^{n,p}$ measured at the same antenna p ($p \in [1, 32]$) at the AP n at the time step t and t', respectively. Quite often the AP can not measure phase values at all antennas during the phase measurement process as discussed in Section 3.3. We normalize the phase correlation by dividing the numerator by c, which is the number of antennas that have phases measured at both time steps t and t'. To ensure the certainty of the phase correlation, we select the phase vector at the latest time step t' in the window such that c is at least 8.

3.6.1.2 Feature Aggregation

The feature aggregation step applies different aggregation operators on the temporal features extracted per AP to generate spatial-temporal features. We describe our aggregation operators below.

• *Average over RSSI feature:* For each RSSI feature except the *Consecutive RSSIs* feature, the operator computes the average of the feature values calculated for the APs having RSSI measurements. For the SD features, the operator considers only the feature values corresponding to the APs that have the highest visibility (*Visibility of AP*). In other words, it discards the feature values corresponding to the APs having

many missing measurements.

• *Pearson correlation over consecutive RSSI:* This operator computes the correlation of *consecutive RSSIs* at multiple APs. When a device is stationary, changes of RSSIs are often similar at most APs, which corresponds to a high correlation value.

• *Max of phase correlations:* This operator computes the maximum of *phase correlation* values computed for multiple APs having phase measurements. We select this operator due to two reasons. First, phase vectors measured at an AP are very sensitive to device motion [31, 72]. Moving the device to another location affects the measured phase vectors significantly, which reduces the phase correlations computed at all APs. Second, when the device is stationary, the APs having line-of-sight (LOS) with respect to the device's location often have stable and high phase correlations. Figure 3.5a compares the CDF of phase correlations computed at LOS APs vs. non-LOS APs when the device is stationary at multiple locations. With the AP density (about 1 AP per 15 m x 15 m) in typical enterprise deployments [20], we expect at least one AP having LOS with respect to a device's location. Figure 3.5b shows the distribution of the max of phase correlations when a mobile device is moving versus stationary.

Before inputting the spatial-temporal features into models that we describe below, we scale all of the features to the range $[0, 1]$. To further address the challenge of missing measurements as described in Section 3.3, if a feature computed from measurements in band A (5 GHz) is missing, we replace it with the feature in band B (2.4 GHz) and vice verse. If the features are missing in both bands, we impute them with the average value of the feature in our training set (mean imputation method [26]).

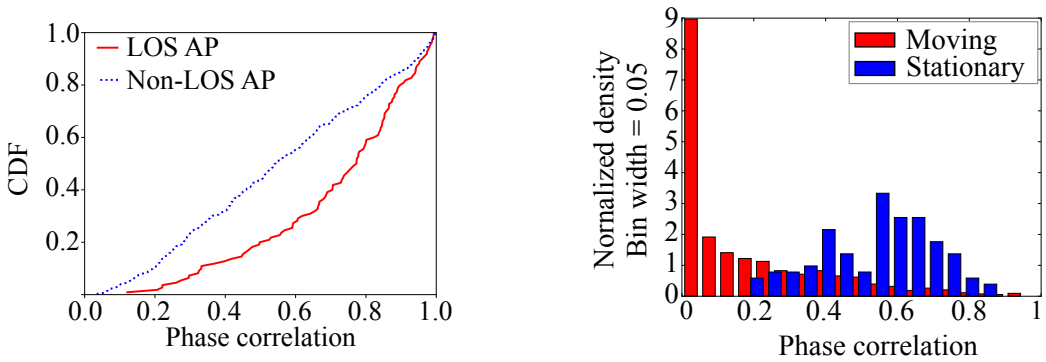


Figure 3.5: (a) CDF of phase correlations at the APs having line of sight (LOS) and non-LOS with respect to device locations (b) Histogram of a max of phase correlations when a device is moving versus stationary

3.6.1.3 Modeling

The modeling step trains different models to learn the relationship between spatial-temporal features and device-motion classification as stationary vs. moving. We considered different models: Recurrent Neural Network (RNN), Random Forest (RF), and Hidden Markov Model (HMM). The purpose of using RNN is to explore if there is still a temporal correlation between the aggregated features.

3.6.2 End-To-End Approach

The goal of the End-To-End (E2E) approach is to train a recurrent neural network (RNN) to classify device motion at time steps having RSSI measurements directly from Wi-Fi measurements (RSSIs and phase vectors), instead of using the designed features from these measurements as described in the previous section. As shown in Figure 3.6, the approach has three models: an E2E-RNN that classifies device motion at the time steps having RSSI measurements, an E2E-RNN that classifies

device motion at the time steps having phase vector measurements, and an RNN that combines the results of these models to classify device motion at the time steps having RSSI measurements. Below, we first describe our design of the E2E-RNN that we use to detect device motion either from RSSI measurements or phase vector measurements. Then, we describe the RNN model for combining the outputs from the two E2E-RNN models.

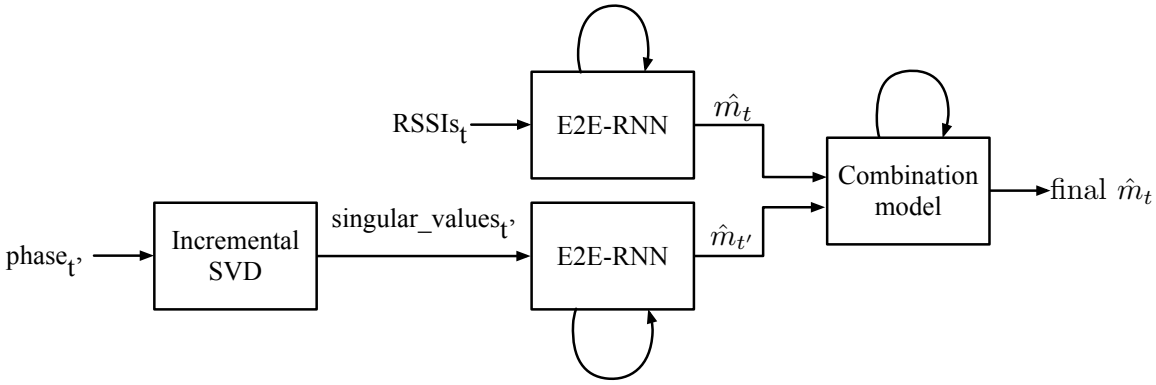


Figure 3.6: End-To-End (E2E) approach

3.6.2.1 End-To-End Recurrent Neural Network (E2E-RNN)

As illustrated in Figure 3.7, we design a neural network that consists of three components. The first component is an RNN of LSTM cells [29] to capture the temporal correlation of the Wi-Fi measurements at each AP. As described in Equation 3.2, the output vector s_t^n of the LSTM cell for the AP n at the time step t is a function of s_{t-1}^n which is the output from the cell in the previous time step at the AP n and a_t^n which is the Wi-Fi measurement at the time step t. The length of the output vector can be optimized by performing a grid search by using a training set. To help the network take into account the missing data, we impute unmeasured data values with zeros and add a binary indicator $mask_t^n$ into a_t^n to indicate if the measurement is completely missing [38].

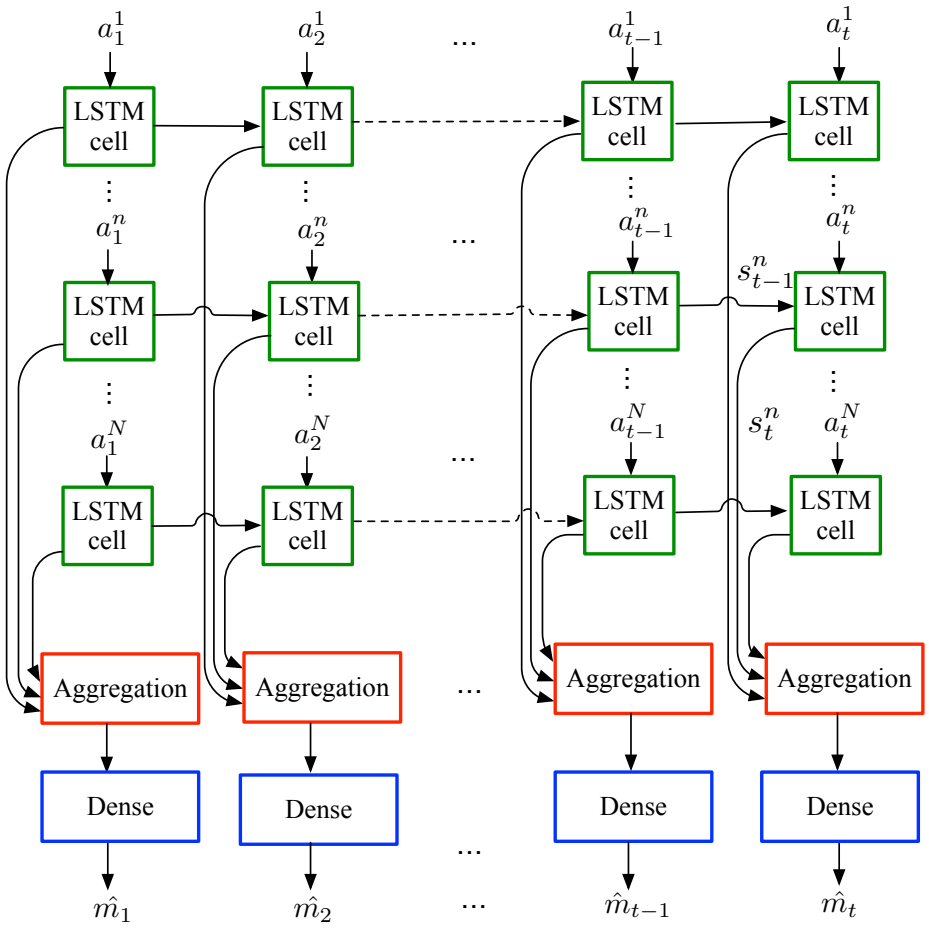


Figure 3.7: End-To-End Recurrent Neural Network (E2E-RNN)

$$s_t^n = f(s_{t-1}^n, a_t^n) \quad \text{where} \quad a_t^n = [\text{mask}_t^n, \text{RSSIs}_t^n, \text{phases}_t^n] \tag{3.2}$$

Given the outputs from N RNNs corresponding to N APs, the aggregation component aggregates the outputs $[s_t^1, ..., s_t^n, ..., s_t^N]$ from the RNNs at each time step. First, we force the network to ignore outputs where the corresponding measurement vector inputs are zero vectors by removing s_t^n if a_t^n is a zero vector for $n \in [1, N]$, and then take the average of the remaining output vectors in each dimension. The output of the aggregation function is a vector s_t.

Given the s_t at the time step t, the dense component [63] outputs the vector $\hat{m}_t$

which consists of the probability of a device's motion at the time step (Equation 3.3).

$$\hat{m}_t = g(s_t) \tag{3.3}$$

We have described our neural network for predicting a device's motion m_t at a time step t. We emphasize that all parameters (weights) in the network are shared across different APs so that the number of parameters in the whole network is small. This approach allows us to train the network with a small training set. We use a gradient descent optimization method that minimizes the loss function that computes average cross-entropies of the predicted motions in a time window [64]. We evaluate this approach in Section 3.7.3.1.

Though it could be possible to train the E2E-RNN that takes both RSSI measurements and phase vector measurements, we train an E2E-RNN to classify device motion for each measurement type. This approach helps to understand the performance of each E2E-RNN for each measurement type, which is not dependent on each other. Below, we describe how we apply the E2E-RNN to classify device motion by using either RSSI measurements or phase vector measurements.

- *E2E-RNN with RSSI measurements:* We train an E2E-RNN model by using only RSSI measurements in band 2.4GHz and 5GHz. In the Equation 3.2, $a_t^n = [\text{mask}_t^n, \text{RSSIs}_t^n]$. In Section 3.7.3, the results show that the model can classify device motion better compared to the feature-based models that use only RSSI features.

- *E2E-RNN with phase vector measurements:* We train an E2E-RNN model for phase measurements to learn temporal patterns across multiple phase vector measurements instead of relying on the phase correlation between a pair of phase vectors, as described in Equation 3.1 in Section 3.6.1.1.

There are two approaches of training the model. The first approach is to use phase vector measurements directly in the input $a_t^n = [\text{mask}_t^n, \text{phases}_t^n]$. Each phase vector consists of phase values. Each value is a complex number. Though it is possible to

train the E2E-RNN with complex numbers [12], the implementation of this approach is challenging. Most deep learning libraries have not fully supported training a deep, complex neural network [12].

In the second approach, instead of using the phase vector measurements directly in the input to the E2E-RNN. We first use Singular Value Decomposition (SVD) to extract singular values from phase vector measurements. To deal with missing phase values in the phase vectors, we impute these values with zeros since these values do not contribute to the result of computing the singular values. In addition, to reduce the cost of computing the singular values, we apply the Incremental SVD (INC_SVD) method [36, 55]. Given the singular values computed based on the current phase vector and the previous phase vectors, we input the values into the network. In the Equation 3.2, $a_t^n = [\text{mask}_t^n, \text{singular˙values}_t^n]$.

3.6.2.2 Combination Model

To generate the final classification results at the time steps having RSSI measurements, we combine the results from the two E2E-RNN models by using an RNN model. Since phase vectors are not measured at every time step, there are missing results from the E2E-RNN model using phase vectors at some time steps. We use the mean imputation method [26] to impute the missing values.

3.7 Evaluation

In this section, we describe how we evaluate our motion detection approach.

3.7.1 Goals and Metrics

We focus on answering the following questions:

1. How do our models classify device motion by using either RSSI measurements

or the combination of RSSI measurements and phase vector measurements?

2. How do our models generalize across different device types and enterprise types?

3. What is the run-time overhead added by performing motion classification?

4. What is the tolerance of the IIPS to misclassification of the models?

Table 3.4 describes our evaluation metrics. These metrics are well-defined in literature, yet we describe them here in the context of device motion classification.

Metric	**Description**
Accuracy	Fraction of correctly classified samples
Precision	Fraction of samples classified as moving which are correct
Recall	Fraction of moving samples correctly classified
F1 score	Harmonic mean of precision and recall
False Negative Rate (Sensitivity)	Fraction of moving samples incorrectly classified as stationary

Table 3.4: Description of metrics used in our evaluation

3.7.2 Data Collection

While a person is carrying a phone and walking on a floor, we record Wi-Fi measurements including RSSIs and phases at APs, the corresponding location estimates computed by the IIPS, and the corresponding actual motion of the device over time. The device's actual motion at time_t is determined by the actual distance that the person moved from time_{t-1} to time_t. As we defined in Section 4.5, if the distance is greater than 1 m, the device motion is moving. Otherwise, it is stationary.

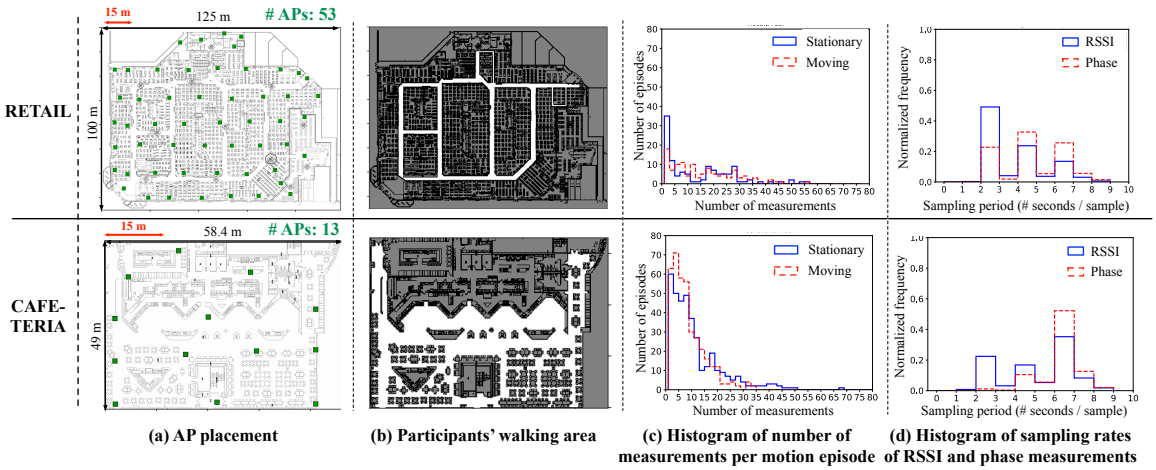


Figure 3.8: (a) AP placements (green squares) at the retail and the cafeteria (about one AP per 15m x 15m) (b) In each floor map, the white color, black color, and dark grey color represent the area that participants visited, obstacles, and the area that the participant did not visit, respectively (c) Histogram of the number of measurements in each motion (stationary or moving) episode. For retail, a large number of episodes have a small number of measurements (less than 3), which makes it difficult to classify device motion. (d) Histogram of the sampling periods (inverse of sampling rate) of RSSI and phase measurements. Phase measurements have smaller sampling rate.

To collect the person's actual locations every second, first, we use a mobile application to mark on the application's floor map the points corresponding to the locations where the person makes a turn or stops moving. Then, given any two consecutive marked points at t and $t+T$ as well as the time stamps associated with these points, we interpolate the person's actual locations every second from t, $t+1$, ..., $t+T$. To ensure the accuracy of the marked points, we require the person to make a turn or to stay at the locations where there are visual landmarks or at intersections. Also, to ensure the accuracy of the interpolated points, we require the person to walk naturally (without changing his or her walking speed).

We performed the data collection during the business hours over multiple days at two different enterprise buildings: a retail store and a cafeteria. We describe our dataset in Figure 3.8 and Figure 3.5. We use the dataset for training and testing our motion detection models as well as illustrating device motion-based use cases.

Enterprise buildings	**RETAIL**	**CAFETERIA**
Traced devices	iPhone 6s, OnePlus	iPhone 6s, iPhone 6s Plus, Nexus 4, Nexus 6
Number of measurements	Moving: 1845, Stationary: 1688	Moving: 3199, Stationary: 4514
Total data collection time (hours)	Moving: 1.6, Stationary: 1.7	Moving: 3.8 Stationary: 6.5
Sampling period (Avg., SD)	RSSI: (3.6, 2.3) Phase: (4.7, 4.9)	RSSI: (5.5, 5.0) Phase: (7.6, 7.1)
Number of devices in the environment	Several to couple hundreds	Several to couple hundreds

Table 3.5: Summary of our dataset

3.7.3 Methodology, Results and Analysis

Below, we present our evaluation methodology, results, and our analysis for each of the goals described in Section 3.7.1.

3.7.3.1 Comparison of motion detection models

We compare the accuracy of different models using the same data set in each enterprise building. Given data collected over multiple experiments, we first randomly permute the experiments performed in each building. Then, we put 80% of the data into a training set for training the models and the remaining 20% of the data for testing the models. To find the best hyper-parameter values of each model, we hold out 10% of

the training data to evaluate the accuracy of the model with a different combination of the hyper-parameter values.

Feature-based approach. Table 3.6 shows the testing results of our models that use either RSSI features or the combination of RSSI features and the phase correlation to detect device motion. We summarize our analysis of the results below.

• The models that exploit the combination of RSSI features and phase correlation outperform the models that only exploit RSSI features in terms of all of the metrics described in Table 3.4. For example, by using RSSI features only, the RF models achieve the accuracy of 0.81 and 0.70 for the cafeteria and the retail, respectively. By using the combination of RSSI features and phase correlation, the RF models achieve the accuracy of 0.87 and 0.77 for the cafeteria and the retail, respectively.

• The RF and RNN models have similar performance. This indicates that using RNN models to further extract temporal correlation of features is not necessary.

• Our models have better motion detection accuracy for the cafeteria than the retail. This is because many motion (stationary or moving) episodes in the retail dataset have a few numbers of measurements (as shown in Figure 3.8c). Hence, given temporally sparse measurements in short periods of stationary or moving, it is more challenging to determine device motion.

To further analyze the contribution of each RSSI feature to the accuracy of our models, we train our models by using each RSSI feature and analyze the performance of our RF models. Table 3.7 shows the accuracy of the RF model when applying each of the features (described in Table 3.3). For the *SD of RSSIs* feature, it gives the best accuracy. For *Visibility of AP* feature, though it gives the worst accuracy compared to other features, it is required by the aggregation operator as described in Section 3.6.1.2. Besides, we want to emphasize that these features achieve much higher accuracy in the prior work [32]. For example, *Visibility of AP* feature achieves

	Models	Accuracy	F1	FNR	FPR	Precision	Recall
Training set (Cafeteria) -> Testing set (Cafeteria)	RNN	0.81, **0.85**	0.78, **0.82**	0.17, **0.17**	0.19, **0.14**	0.74, **0.80**	0.83, **0.84**
	RF	0.81, **0.87**	0.77, **0.84**	0.22, **0.13**	0.16, **0.13**	0.76, **0.81**	0.78, **0.87**
	HMM	0.78, **0.82**	0.76, **0.79**	0.16, **0.17**	0.26, **0.19**	0.69, **0.75**	0.84, **0.83**
	E2E	0.83, **0.88**	0.78, **0.85**	0.24, **0.11**	0.13, **0.13**	0.80, **0.82**	0.76, **0.89**
Training set (Retail) -> Testing set (Retail)	RNN	0.71, **0.79**	0.74, **0.80**	0.18, **0.14**	0.41, **0.29**	0.67, **0.75**	0.82, **0.86**
	RF	0.70, **0.77**	0.72, **0.78**	0.24, **0.19**	0.36, **0.27**	0.68, **0.76**	0.76, **0.81**
	HMM	0.72, **0.74**	0.73, **0.74**	0.24, **0.25**	0.33, **0.27**	0.70, **0.74**	0.76, **0.75**
	E2E	0.71, **0.66**	0.73, **0.69**	0.25, **0.27**	0.32, **0.41**	0.71, **0.65**	0.75, **0.73**

Table 3.6: Testing results of models trained by using data collected at the same enterprise type. For each metric, the first number is the result of the model using only RSSI features, the second number (**bold**) is the result of the model using both RSSI features and phase feature. For feature-based approaches (RNN, RF, and HMM), the models that use both features outperform the models that use only RSSI features. For the E2E approach, given the sufficient training set collected at the cafeteria, the approach achieves better performances compared to the feature-based approaches. However, given a small training set collected at the retail, the E2E approach has lower performances.

the accuracy of 0.86, but only 0.63 with our dataset. It is probably because the sampling rate of RSSIs in their setups is periodic and doubles the sampling rate of RSSIs in our setups (Section 3.4).

End-to-end (E2E) motion detection approach. We implement our E2E motion detection models described in Section 3.6.2 by using the TensorFlow library [62]. In the training process, we find the best hyper-parameter values by performing a grid search. In particular, the number of hidden units in a LSTM cell is 6. To train the E2E models, we use the Gradient descent optimization method [64]. Below, we describe our results and analysis of the models in the E2E approach: the E2E-RNN with RSSI measurements, the E2E-RNN with phase vector measurements, and the

Feature-based approach using RSSI measurements		Cafteria		Retail	
RSSI features	**Aggregation operator**	**Accuracy**	**F1**	**Accuracy**	**F1**
SD of RSSIs	Average	0.80	0.76	0.69	0.7
SD of RSSI differences	Average	0.76	0.68	0.66	0.69
Consecutive RSSI difference	Average	0.75	0.69	0.66	0.69
Consecutive RSSIs	Pearson correlation	0.73	0.64	0.59	0.64
Visibility of AP	Average	0.61	0.37	0.64	0.63

Table 3.7: Motion classification accuracy of feature-based approach when each RSSI feature is applied separately

combination model.

• *E2E-RNN with RSSI measurements:* In Table 3.6, for the testing set collected at the cafeteria, the E2E-RNN taking only RSSI measurements achieves accuracy of 0.83 and F1 score of 0.78. For the testing set collected at the retail, E2E-RNN achieves accuracy of 0.71 and F1 score of 0.73. These results show that the E2E approach can achieve better motion classification accuracy compared to the feature-based RF model using only RSSI measurements.

• *E2E-RNN with phase vector measurements:* Table 3.8 shows the results of using only phase vectors to perform motion detection with two different approaches: (i) the phase correlation and (ii) the combination of the incremental SVD (INC_SVD) and the E2E-RNN model. The latter approach achieves better motion classification accuracy with the testing set collected at the cafeteria. However, for the testing set collected at the retail, the latter approach has a lower F1 score compared to the first approach that uses the phase correlation. The reason could be the training set collected at the retail is insufficient to train the model. As shown in Figure 3.8 and Table 3.5, the data collection time at the retail is significantly smaller compared to that of the cafeteria. Also, the sampling rate of phase vector measurements is lower compared to that of RSSI measurements.

• *Combination model:* In Table 3.6, for the testing set at the cafeteria, our E2E

	Phase features	Accuracy	F1	FNR	FPR	Precision	Recall
Training set (Cafeteria) -> Testing set (Cafeteria)	Phase correlation	0.84	0.76	0.19	0.15	0.71	0.81
	E2E-RNN-phase	0.88	0.82	0.15	0.10	0.79	0.85
Training set (Retail) -> Testing set (Retail)	Phase correlation	0.73	0.73	0.25	0.30	0.71	0.75
	E2E-RNN-phase	0.73	0.69	0.37	0.19	0.76	0.64

Table 3.8: Motion classification accuracy achieved by using only phase vector measurements with two approaches: (i) the phase correlation (ii) the combination of the Incremental SVD (INC_SVD) and the E2E-RNN, called E2E-RNN-phase.

approach that consists of E2E-RNN models taking both RSSI measurements and phase vector measurements achieves accuracy of 0.88 and F1 score of 0.85, which is slightly better than the feature-based approaches. However, for the testing set at the retail, the combination model achieves much lower accuracy compared to the accuracy of the feature-based approaches. The insufficient amount of training data collected at the retail can be the root cause of the low accuracy of the E2E-RNN models and the combination model.

3.7.3.2 Model generalizability

To investigate the generalizability of our feature-based models across enterprise types and device types, we train our models by using data collected at the cafeteria and test the models by using data collected at the retail, and vice versa. Table 3.9 shows the accuracy of our models that use either RSSI features or the combination of RSSI features and phase correlation for classifying device motion.

- *Feature-based approach:* Compared to the testing results of the RF models in Table 3.6, the RF models achieve very similar accuracy in terms of all of the metrics (as described in Table 3.4) though each RF model is trained by using data collected in another building.

• *E2E approach:* Compared to the testing results of the E2E approach in Table 3.6, the E2E approach has much lower accuracy in terms of all of the metrics (as described in Table 3.4). These results indicate that the E2E approach can be over-fitted to the dataset collected in each environment. In the future work, we plan to combine data set collected across various environment settings to train the E2E models.

We conclude that the RF models are more generalized across enterprise types and device types.

	Models	Accuracy	F1	FNR	FPR	Precision	Recall
Training set (Retail) -> Testing set (Cafeteria)	RNN	0.78, **0.78**	0.68, **0.67**	0.41, **0.43**	0.10, **0.08**	0.80, **0.83**	0.60, **0.57**
	RF	0.83, **0.87**	0.79, **0.84**	0.20, **0.13**	0.15, **0.13**	0.78, **0.81**	0.81, **0.87**
	E2E	0.43, **0.59**	0.51, **0.56**	0.26, **0.27**	0.79, **0.41**	0.39, **0.65**	0.75, **0.73**
Training set (Cafeteria) -> Testing set (Retail)	RNN	0.70, **0.73**	0.70, **0.73**	0.31, **0.30**	0.29, **0.24**	0.71, **0.75**	0.69, **0.70**
	RF	0.72, **0.78**	0.75, **0.79**	0.19, **0.18**	0.37, **0.26**	0.69, **0.77**	0.81, **0.82**
	E2E	0.53, **0.75**	0.18, **0.75**	0.90, **0.26**	0.03, **0.25**	0.76, **0.76**	0.1, **0.74**

Table 3.9: Testing results of feature-based models trained by using data collected at the retail and tested by using data collected at the cafeteria, and vice versa. Compared to Table 3.6, the performance of the RF models are slightly different though the RF models, in this case, are trained by using data collected in another building.

3.7.3.3 Computation time

To make the IIPS scalable, the running time of detecting a device's motion at a time step needs to be significantly lower than the running time of estimating the device's location. To investigate the overhead added by performing motion detection, we compare the average running time of our motion detection approaches and the localization method (described in Section 2), at each time step. We measure these running times on the same computer (Intel CPU @ 3.70GHz).

For the feature-based RNN model, extracting six aggregated features from 13 APs and performing the model take about 1.3 ms and 0.7 ms, respectively. Thus, at a time step, the approach takes about 2 ms. The total running time can be reduced by extracting the features from the APs reporting measurements, instead of from all APs within an environment as in our current implementation.

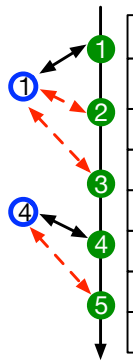
For the E2E approach, the computational cost consists of the running time for performing inferences with the two E2E-RNN models and the combination model as well as performing INC_SVD. Each E2E-RNN model takes about 4 ms. The combination model takes about 1 ms. The INC_SVD takes about 11 ms. Therefore, at a time step, the E2E approach takes about 20 ms. Similar to the implementation of the feature-based approach, the total running time can be reduced significantly by considering only APs reporting measurements.

The localization method takes about 15 ms for computing a device's location by using measurements from 6 APs reporting measurements. The running time is proportional with the number of APs. The running time of the feature-based approach is only 13% of the running time for computing device computation at a time step. Therefore, for a device correctly predicted as stationary, we can save about 87% of the CPU time spent on each location computation.

3.7.3.4 Tolerance to misclassification

We discuss the impact of each misclassification type, namely, false positive rates (actual stationary, predicted as moving), and false negative rates (actual moving, predicted as stationary), respectively. False positives (FPs) have no impact on location accuracy as the location computation is triggered for the sample. On the other hand, false negatives (FNs) may impact location accuracy. In Figure 3.9, we depict how FNs impact accuracy. The estimated location for act_2, act_3, will be the *last*

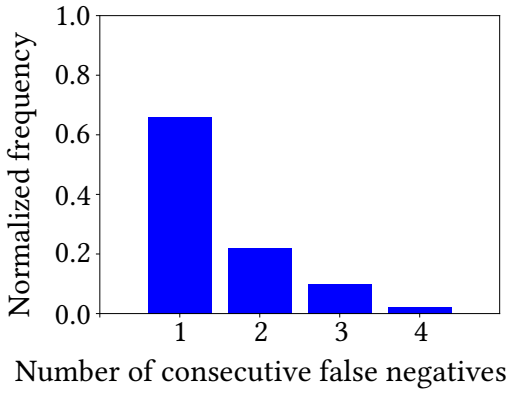
computed location (=est_1), and for act_5 it will be est_4. As a result, if the number of consecutive FNs is large, it may have a significant impact on location accuracy.



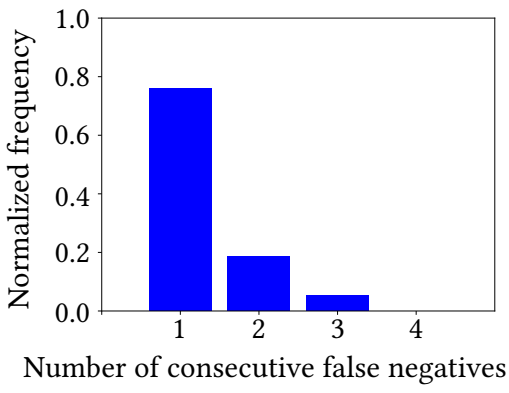
Actual		Motion-based computation saving		
Location	**Motion**	**Prediction**	**Computed**	**Est. loc.**
act_1	moving	moving	✓	est_1
act_2	moving	stationary	✗	est_1
act_3	moving	stationary	✗	est_1
act_4	moving	moving	✓	est_4
act_5	moving	stationary	✗	est_4
…	…	…	…	…

Figure 3.9: Motion-based computation saving

Figure 3.10 shows that we rarely have more than 3 consecutive FNs and 70% of FNs are not consecutive. In addition, Figure 3.11 shows that the FNR of our RF model (about 0.16 on average) has a negligible impact on the location accuracy as the CDFs of location errors when a device is moving with and without skipping location computations mostly overlap.

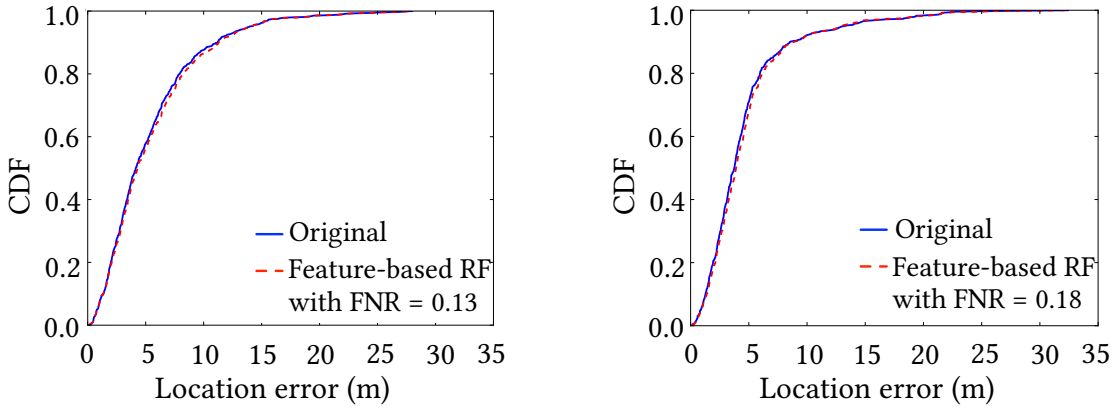


(a) Cafeteria



(b) Retail

Figure 3.10: Histogram of number of consecutive false negatives



(a) Cafeteria

(b) Retail

Figure 3.11: CDF of location accuracy of moving device with and without skipping location computation when device motion is predicted as stationary

3.8 Motion-based Enhancements

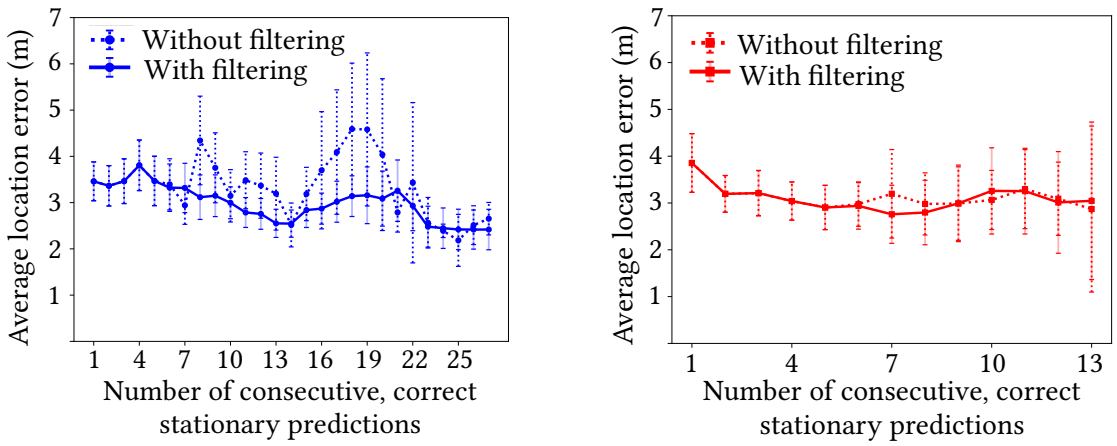
In this section, we demonstrate three important use cases of motion classification models to enhance LT accuracy and scalability of the IIPS as described in Section 4.1.

3.8.1 Motion-based Computation Saving

Towards achieving high scalability, proactive location computation savings can be achieved by only computing location when a device is classified as moving. Thus, in an enterprise IIPS setup, which tracks thousands of devices, depending on the fraction of tracked devices that are stationary, a significant computation saving can be achieved. Computation savings is directly proportional to the *true negative rate* (TNR) of the motion classifier. Tables 3.6 and 3.9 list the FPR of our proposed motion classifiers. For the motion classifiers using RSSIs and phases, TNR = (1 - FPR) $\simeq$ 80%, i.e., 80% of stationary points can be correctly identified for computation

saving. The potential risks of misclassification are discussed in Section 3.7.3.4 where we establish that the impact of misclassification is not significant. Further, a variety of computation saving schemes can be realized ranging from most aggressive (not re-compute for devices classified as stationary) to conservative ones (selectively re-compute if a device is classified as stationary yet skip computation for longer than certain duration).

3.8.2 Motion-based Filters



(a) Cafeteria

(b) Retail

Figure 3.12: Motion-based filtering to improve accuracy.

By leveraging motion detection, location accuracy can be improved by applying certain filters such as Particle Filter or Kalman Filter [65]. In particular, motion classification based filter can be applied to the location estimates to mitigate the errors. Further, different filters can be applied for stationary vs. moving. To demonstrate this idea, we perform motion classification over our collected data sets. For simplicity, we apply a cumulative median filter for stationary devices. Fig. 3.12 shows the average of location errors with and without applying the filter on different numbers

of consecutive location estimates that are correctly classified as stationary. The error bars represent the 90th percentile confidence intervals of the corresponding average errors. For the cafeteria (Fig. 3.12b), the filtered location estimates are much less scattering and closer to the corresponding actual locations. For the retail, the location accuracy with and without applying the filter are similar. The reason could be the number of consecutive stationary predictions is smaller and participants are actually stationary for smaller periods of time (Fig. 3.8c).

3.9 Discussion

In this section, we compare the feature-based approach and the end-to-end (E2E) approach in Table 3.10 and describe directions for future research that build upon this work.

• The generalizability of the E2E approach can be improved by combining training data collected across various environment settings.

• The current implementation of the E2E approach computes features for all APs deployed on a floor. The computational cost of this approach can be reduced significantly by considering only APs reporting measurements.

• In the E2E approach, we use the incremental SVD method to extract singular values from phase vectors consisting of complex phase values, and then use these values as inputs into the E2E-RNN model. The main limitation of the incremental SVD method is that it requires specifying a forgetting factor, which controls the effect of previous measurements on its current output [55]. Future work could consider applying a deep complex network [12] that can learn temporal features directly from the phase vectors without specifying the forgetting factor.

• Classifying device motion in real time can be further improved by considering the frequency with which a device was stationary or moving in the past.

	Feature-based approach	**E2E approach**
Motion classification accuracy	This approach achieves high classification accuracy.	This approach achieves slightly higher accuracy. The E2E approach can learn temporal-spatial features directly from a training dataset effectively.
Model generalizability	This approach is generalized.	This approach is less generalized. It indicates that the E2E approach can be over-fitted to the dataset collected in each environment.
Computational cost	This approach has negligible computational cost.	This approach has significant higher computational cost, which is slightly higher than the phase-based AoA localization method.
Training dataset	This approach requires a small amount of training data (collected in couple of hours)	This approach requires a larger amount of training data (collected in couple of days)

Table 3.10: Summary and comparison of our motion detection approaches: the feature-based approach and the end-to-end (E2E) approach

• The use cases of computation saving and location accuracy improvement were presented orthogonally. A possible future improvement is to develop a model to simultaneously achieve both.

3.10 Conclusion

In summary, we have shown that MotionScanner can exploit the temporal patterns of noisy, temporally sparse, and partial measurements from the IIPS to detect device motion accurately. We also demonstrated that MotionScanner can enhance

the performance of the IIPS in terms of scalability and location accuracy. We envision that MotionScanner can enable other interesting use cases for enterprises such as client behavior analytics and regions of interest based on how long and how frequent user devices stay or move in particular areas.

4 Infrastructure-based Map Matching

This chapter describes our second contribution that is the infrastructure-based map matching to exploit pre-defined path segments in a floor map to improve localization and tracking. We design, implement, and evaluate two map matching approaches: a graph-based approach and an image-based approach. The graph-based approach represents the floor map as a graph. The novelty of this approach lies in applying geometric and topological constraints based on our human mobility study to select which path segment to align the current location estimate. The novelty of the image-based approach lies in representing input data including location estimates and the floor map as 2D images. This representation enables developing encoder-decoder neural networks to exploit spatial relationships in input images to potentially improve location accuracy.

4.1 Motivation

Achieving accurate localization and tracking of mobile devices is essential to enable location-based services such as navigation and proximity marketing. However, due to noisy Wi-Fi measurements obtained by the IIPS, location estimates are often scattering around the corresponding actual locations. As a result, location estimates can be aligned on obstacles such as walls and furniture when a user is moving, which strongly impacts the quality of the location-based services. With the scattering location estimates, it is difficult to determine the path a user is currently on and provide

the turn-by-turn navigation service to the user. Moreover, multiple sales information available at stores on both sides of a hallway can be delivered to a user when he or she is walking through the hallway without visiting these stores.

To further improve the localization and tracking accuracy of the IIPS, we make several observations. The IIPS often store a digital floor plan representing obstacles on the floor. Operators of enterprise buildings can define the paths that their customers can take on the floor plan. Therefore, we can exploit the pre-defined paths to constrain location estimates on these paths to improve the tracking accuracy.

In this chapter, we address the *online map matching* problem, i.e., given location estimates generated by the IIPS and a floor map consisting of pre-defined path segments, how to align the current location estimate on the path taken by a user and close to the user's current location. Below, we describe the challenges that we address to solve the problem.

4.2 Challenges

Figure 4.1 describes an example of the online map matching problem with location estimates generated by the IIPS. Map matching faces a quartet of challenges:

- First, location estimates are often scattering around the corresponding actual locations due to noisy Wi-Fi measurements obtained by the IIPS. Therefore, it is challenging to determine which path segment a user is currently on.

- Second, location estimates are temporally sparse and non-periodic because the IIPS obtain Wi-Fi measurements in a best effort manner as described Section 2.3. Therefore, there are weak correlations between consecutive location estimates, which in turn makes it challenging to exploit historical location estimates to solve the problem.

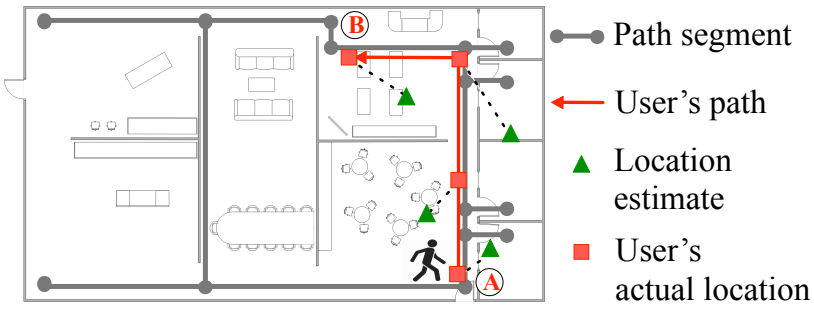


Figure 4.1: A user walks from location A to location B on the floor map. Given the floor map composed of the path segments and the sequence of time-stamped location estimates, the map matching problem is to constrain the current location estimate on the path taken by the user and close to the user's current location.

- Third, floors within large enterprise buildings often have many possible walking paths. Therefore, it is challenging to determine which path the user is currently on.

- Fourth, due to the latency incurred in transmitting Wi-Fi measurements to a server of the IIPS and in the location estimation process, the location estimates often lag the corresponding actual locations. Even though a location estimate is aligned on the path segment taken by a person, this path segment is not the current path segment he or she is currently on. This challenge makes measuring map matching accuracy by using existing metrics less suitable.

4.3 Contributions

We design, implement and evaluate the two infrastructure-based map matching approaches to constrain location estimates on the path taken by a user and close to the user's current location. Below, we describe our main contributions in these approaches: a *graph-based approach* and an *image-based approach*.

- *Graph-based approach*: The key idea of this approach is to represent a floor map as a graph and apply geometric and topological constraints to find the correct path segment currently taken by a user and then align the current location estimate on the selected path segment. Compared to the existing graph-based approaches, we make the following contributions:

  - We performed a preliminary human mobility study in six different indoor environment settings to study how frequently a person make a turn. Based on the result of this study, we proposed a number of turns constraint and applied it to solve the map matching problem.
  - We developed a Hidden Markov model that applies a combination of several constraints including a point-to-curve constraint, a travel distance constraint, and a number of turns constraint to select the correct path segment.
  - To evaluate this approach, we collected data in two different environment settings (a workplace and a lab). We also proposed a path-alignment accuracy metric. The metric measures the percentage of location estimates that are aligned on the path taken by a user and lag the corresponding actual locations. Our results showed that our approach can improve path-alignment accuracy, compared to prior graph-based approaches. About 40% of location estimates are now aligned on the path taken by a user and 4 meters behind the corresponding user locations.

- *Image-based approach*: The key idea of this approach is to represent input data consisting of both a floor map and a location estimate as 2D images. This *novel representation* enables combining these images and developing deep learning neural networks to learn constraints for performing map matching. To the best

of our knowledge, none of the prior work has explored this direction. Below, we describe the input representation and our investigation on how our image-based approach solves the map matching problem.

- To take into account the constraint that a user's location is on a pre-defined path segment, we multiplied a floor image representing path segments and a location map representing a sampled 2D Gaussian distribution of a location estimate on the floor. We called the output of this element-wise multiplication a spatial image. Given a sequence of spatial images corresponding to a sequence of location estimates until the current time, we developed a 3D convolution encoder-decoder neural network to minimize the average of the location estimation errors.
- To evaluate this approach, we developed a simulation to generate various datasets by varying location estimation errors and temporal distances of location estimates. Our results showed that, while the graph-based approach does not improve location accuracy, the image-based approach can significantly improve location accuracy. When the distances between consecutive location estimates are about five meters, the image-based approach reduces the median of the estimation errors from 5.3 m to 4.0 m. However, compared to the graph-based approach, the image-based approach aligns a smaller percentage of location estimates on the path taken by a user. This is because the image-based approach does not strictly constrain the location estimates on the path segments.

4.4 Related Work

There are two important aspects of map matching algorithm: map representations and constraints established based on a map presentation to perform map matching.

Below, we describe prior effort in each of these aspects.

4.4.1 Map Representations

Most prior works represented a set of pre-defined paths in an environment either as a set of arcs or as a graph [6, 7, 35, 42, 47, 50, 52, 59, 75]. An arc consists of piece-wise linear path segments. Given this representation, geometric constraints can be established as we describe in Section 4.4.2.1. A graph consists of vertices and edges. Each vertex represents a coordinate of an end point of a path segment (or an arc). Each edge represents a path segment (or an arc). Given the graph representation, topology of path segments (or arcs) can be further exploited to establish topological constraints 4.4.2.2. In our work, we explore the graph representation and propose a new topological constraint that helps improve the path-alignment accuracy of our map matching algorithm. In Section 4.6, we describe the graph representation in detail.

In addition, we further explore an image representation of input data consisting of pre-defined path segments and a sequence of location estimates. This representation can enable leveraging deep neural networks such as Convolutional Neural Networks (CNN), which have been applied successfully for understanding and making inferences with various data such as images and video having Euclidean structures [9]. The novelty of this approach is the new presentation of the input data, which has not been explored in the prior work. We represent the pre-defined path segments on a floor as an image and a sequence of location estimates as images. This representation enables us to explore whether a deep neural network can be developed to address the map matching problem. In Section 4.7, we describe the image representation in detail.

4.4.2 Graph-based Constraints

With the graph representation, prior work proposed several constraints to address the map matching problem. Our goal is to find constraints that can be applied to IIPS data for robust map matching. This section examines constraints proposed in prior work and applied in existing map matching approaches, for location estimates generated by both indoor and outdoor positioning systems.

4.4.2.1 Geometric Constraints

Point-to-curve proximity constraint [6]: Bernstein and Kornhauser first proposed a point-to-curve proximity constraint for outdoor map matching. The road network is represented as a set of curves wherein each curve comprises a sequence of path segments. The constraint implies that the closer a location estimate is to a path segment, the more likely it is to get aligned on that path segment. The metric for closeness is the distance between the location estimate and its orthogonal projection on the path segment. This constraint is not robust when the location estimates are very noisy.

Curve-to-curve proximity constraint [75]: A trajectory curve is a set of straight lines connecting the current location estimate and k previous location estimates where the k^{th} previous estimate is highly likely to be on a small set of candidate road curves. After projecting the trajectory curve on each of these candidate curves, the more similar the length of the trajectory curve to the length of a projected trajectory curve, the more likely the current estimate is aligned on the projected trajectory curve. This constraint is more robust than the point-to-curve proximity constraint. Given noisy, infrequent location estimates and dense road networks, consecutive location estimates often need to be aligned on multiple curves. Since this constraint is not well-defined

in our case, we do not apply it in our map matching algorithm.

Orientation (direction, bearing) constraint [7, 52, 59]: The smaller the angle between a user's heading and a path segment, the more likely the location estimate is to get aligned on the path segment. The user's heading can be either obtained from the inertial sensors on the user's device or estimated as the vector between the previous location estimate and the current estimate. The heading estimate is often not accurate with noisy, infrequent location estimates. Because the IIPS track user location without collecting inertial sensor data, we can not apply the orientation constraint in our work.

Same curve constraint [42]: Mohamed *et al* proposed a same curve constraint based on the observation that mobile users often travel on the same road rather than make frequent turns in a short time. The current location estimate is more likely be aligned on the same curve as the previous location estimate. For indoor environments having dense curves, a mobile user can walk on different curves during the time between two consecutive location estimates. Therefore, this constraint could not be directly applied for the location estimates from the IIPS. In our work, we performed a human mobility study on three different building floors to extract the turn-making frequencies of participants over different walking distances. Our observation is that mobile users often make a small number of turns relative to the travel distance, instead of staying on the same curve.

4.4.2.2 Topological Constraints

Travel time constraint [35]: Krumn *et al* proposed a travel time constraint based on the observation that the actual travel time between two consecutive location estimates is similar to the estimated travel time between the corresponding actual locations of these estimates. The current location estimate is aligned on the path segment such

that the estimated travel time is most similar to the actual travel time. To estimate the travel time, the authors assume that a mobile user travels on the shortest path between these actual locations, and take into account the speed limits of roads. For indoor environments, there are no speed limits. Moreover, the IIPS do not collect inertial sensor data from the user's device, therefore, estimating the user's speed at any given time has high uncertainty. In our work, we use the average walking speed of participants as the estimated walking speed.

Travel distance constraint [47]: Newson *et al* proposed a travel distance constraint based on the observation that the Euclidean distance between two consecutive location estimates is similar to the actual travel distance between the actual locations corresponding to these estimates. With the shortest path assumption, the current location estimate is aligned on the path segment such that the shortest path between the current and the previous aligned location estimates is most similar to the Euclidean distance between these location estimates. This constraint is more robust than the travel time constraint since it is independent to a user's speed. In our work, we show that applying this constraint and a number of turns constraint described below gives the best map matching accuracy.

Number of turns constraint [50]: Osogami and Raymond proposed the number of turns constraint based on the heuristic that mobile users often make a small number of turns while traveling in outdoor environments. Essentially, they proposed a cost-metric for a path which considers both the path's length and the number of turns in the path to find the least-cost path between two consecutive location estimates. From our preliminary study of human mobility in indoor environments, we observed that the number of turns that people make over different distances across different building floors are similar and relatively small. Therefore, we formulate and apply this constraint but in a different way from the authors' proposal.

4.4.2.3 Semantic Constraints

Major curve constraint [42]: Mohamed *et al* proposed a major curve constraint based on the observation that mobile users often travel on major curves rather than on secondary curves. The rank of each curve is extracted from an open database (e.g. OpenStreetMaps), or estimated from the road type (freeway, side roads, etc.). For indoor environments, this observation can help improve the map matching accuracy. However, as estimating the path rank in an indoor environment requires a large data set, we will consider this constraint in future work.

In other work, Aly and Youssef [2] used semantics of a curve such as a bump, bridge, tunnel, and turn restrictions as constraints for map matching. Utilizing inertial sensor data from a user's device, the algorithm predicts the curve semantics, and compares it to the actual semantics of different curves to align the location estimates on the curve with similar semantics. However, as the IIPS do not collect phone data, we do not apply semantic constraints.

4.5 Problem

In this section, we describe the online map matching problem, stating key definitions first.

4.5.1 Definitions

- A *path segment* is a straight line connecting two locations on a floor. It has no intersections with other path segments except at its ends. Figure 4.1 shows examples of path segments.

- A *floor map* consists of a set of predefined path segments. The floor map can be represented as a graph (Section 4.6) or as an image (Section 4.7).

- A *location estimate* r_i at time step t_i is the pair of coordinates on a two dimensional floor plan.

- A *user's trajectory* TR_n at the current time step t_n is a sequence of time-stamped location estimates until t_n.

$$TR_n = \{(r_1, t_1), ..., (r_i, t_i), ..., (r_n, t_n)\}$$

4.5.2 Problem Statement

Given a floor map and a user's trajectory TR_n until the current time step t_n, how to constrain the current location estimate r_n on the path taken by the user and close to the user's current location. Figure 4.1 shows an example of the problem.

4.6 Graph-based Approach

As described in Section 4.4.1, most prior works use a graph to represent a floor map which consists of path segments as described in Section 4.5. In our graph-based approach, we represent the floor map as an undirected graph $G(V, E)$ which consists of the edge set E representing the sets of path segments and the vertex set V representing the segment endpoints. The key idea of this approach is first to select a correct path segment that the user is currently on, and then align the current location estimate on the selected path segment.

Our online graph-based map matching algorithm consists of four steps, shown in Figure 4.2. The smoothing step smooths location estimates in a user's trajectory to reduce noise. The modeling step applies constraints from the user's trajectory, a path network, and a human mobility model to build a Hidden Markov Model (HMM). Given the model, the path segment selection step uses a Viterbi decoder to solve the HMM to get a path segment for aligning the current location estimate. The alignment

step aligns the estimate to its nearest point in the selected segment [6, 35, 47, 75]. Below, we first describe our human mobility study. Then, we describe the steps of our map matching algorithm.

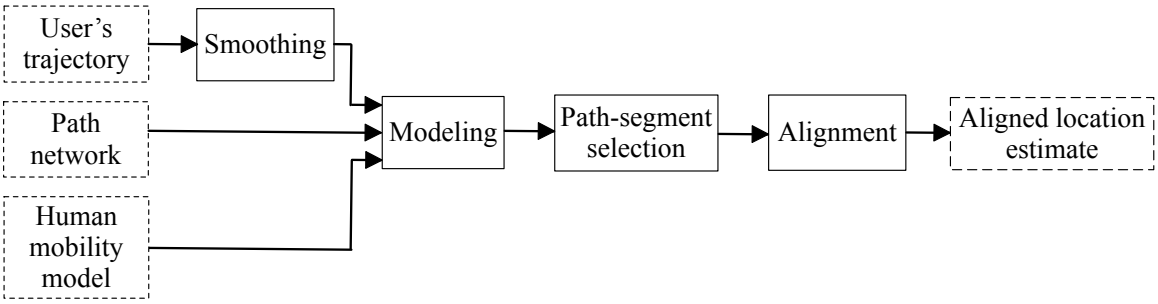


Figure 4.2: Our map matching algorithm has four steps: smoothing, modeling, path segment selection, and alignment.

4.6.1 Human Mobility Study

For outdoor map matching, prior work has proposed the heuristic that people often make a small number of turns relative to their traveling distances [50]. However, in outdoor environments, the heuristic has not been studied thoroughly. The purpose of our study is to determine if a number of turns constraint based on this heuristic can be applied to perform map matching robustly in different indoor environments. Below, we describe our data collection, sampling method, and analysis.

Questions	**Choices**		
Are you familiar with this floor?	No	Quite	Very
Do you have a clear walking direction?	No	Quite	Very
What is your visiting purpose?	Browsing	Finding	

Table 4.1: Questionnaire for studying human mobility

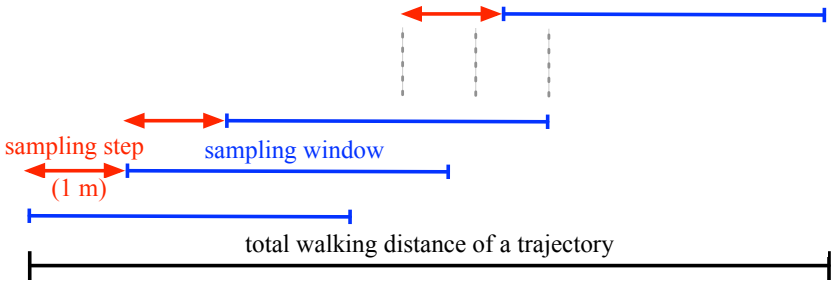


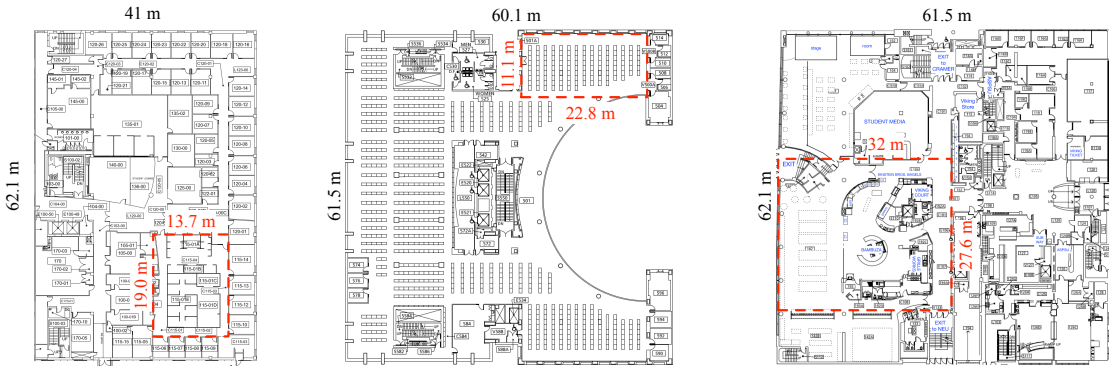
Figure 4.3: Sampling the number of turns in each sampling window moved from the beginning of a participant's trajectory. The sampling step, i.e., the distance between two consecutive sampling windows is one meter.

4.6.1.1 Data Collection

In our experiments, we collect actual locations and associated time-stamps for a group of six adults as they walk on three different building floors (workplace, library, and cafeteria) including the subareas shown in Figure 4.4. We select these building floors and subareas because they have different settings and dimensions.

In the experiment, we ask each participant to first answer a questionnaire described in Table 4.1 and then walk on a floor for *browsing* (exploring the floor), *finding* a place (room), or an object (book) on the floor. Each participant performs two pairs of browsing and finding experiments per floor and one pair of these experiments per subarea. For browsing, we require participants to walk at least two minutes on each floor and at least one minute on each subarea.

We develop a mobile application that collects a participant's questionnaire response. It lets the participant indicate mobility (stationary, moving, or making a turn) during the experiment. To collect the actual participant locations, we walk with the participant to mark on a digital floor map a point (location) corresponding to when the participant indicates his or her mobility. Also, we require each par-



(a) Workplace (b) Library (c) Cafeteria

Figure 4.4: Floor plans of (a) workplace, (b) library and (c) cafeteria where we perform our experiments. Areas surrounded by dashed red lines are subareas considered as a small floor in our experiments. In total, we perform six sets of experiments for six different areas.

ticipant to walk normally (not dramatically altering speed between two consecutive indications), so that we can interpolate the actual locations in between every second.

4.6.1.2 Sampling

We use a sliding window method to sample the number of turns for different walking distances. As shown in Figure 4.3, we slide the window from the beginning of the trajectory, extracted from an experiment, to calculate the number of turns in each window. The window size is the walking distance. We repeat this sampling process with different window sizes ranging from 1m to max_wnd m. The distance between two consecutive windows is a sampling step, chosen in our study to be 1m. We choose max_wnd as the maximum walking distance between two consecutive IIPS-generated location estimates as our purpose is to apply the number of turns constraint to compute the transition probability between two path segments to align these estimates.

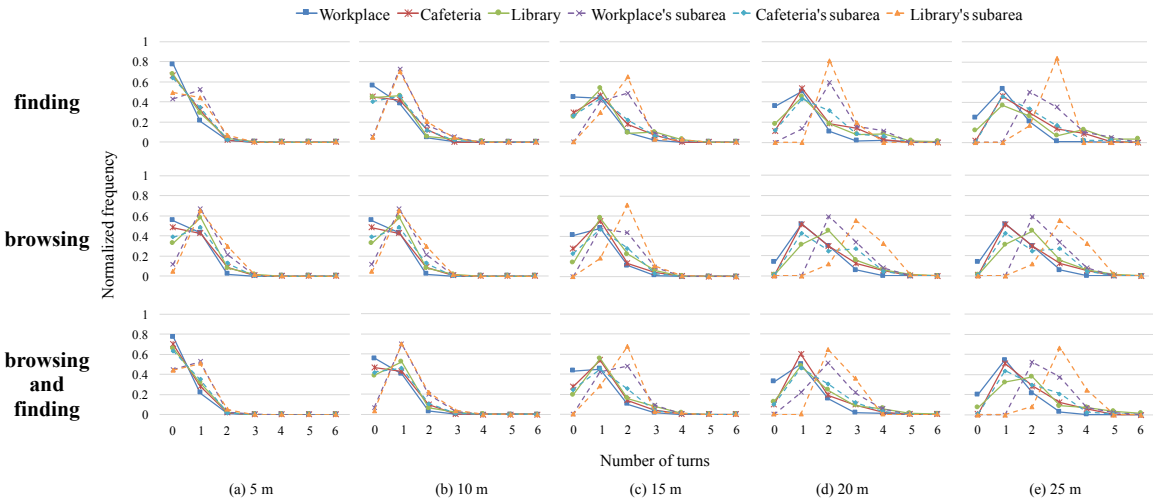


Figure 4.5: Each row shows the distributions of the number of turns (samples) that participants make after walking (a) 5 m, (b) 10 m, (c) 15 m, (d) 20 m and (e) 25 m on six different areas described in Fig. 4.4 for finding (first row) or browsing (second row). The third row shows the distributions of number of turns when combining the samples in both cases (equal number of samples in each case). The sample distributions in large areas (workplace, cafeteria, library, and cafeteria's subarea) are similar regardless of their different settings, dimensions, and participant's visiting purposes (finding or browsing). For small areas (workplace's subarea and library's subarea), the sample distributions are different from those in large areas and vary slightly when the walking distance increases.

4.6.1.3 Analysis

Figure 4.5 shows the distributions of the number of turns that participants make over different walking distances for six different areas (see Figure 4.4). From the plots in the first two rows, we make several observations:

- The number of turns taken is relatively small. It increases only slightly with the walking distance. When browsing or finding, participants make at most

five turns after walking 25 m. For large areas (workplace, cafeteria, library and cafeteria's subarea), participants often make very few turns (e.g., only one turn after walking 25 m). For small areas (workplace's subarea and library's subarea), the number of turns increases slightly faster with the walking distance since most of path segments in a small area are short, which prevents participants from selecting long paths to make fewer turns.

- For large areas, the distributions of the number of turns are similar regardless of the areas' settings, dimensions, and participant's walking purposes (finding or browsing). For small areas, the sample distributions are different from those in large areas and vary slightly when the walking distance increases.

From the observations above, we hypothesize that the number of turns constraint could be applied to set a weight for a predicted path (the shortest path) between two consecutive aligned location estimates on a floor map. The last row in Figure 4.5 shows the distribution of number of turns at different distances obtained by combining the samples in both cases (browsing and finding) with the same number of samples in each case. We notice the same observations as before. Therefore, we can use the distributions of the number of turns in these large areas (or small areas) to set weights for predicted paths in an arbitrary large area (or an arbitrary small area). We expect that this method may not perform well in small areas since the distributions vary between small areas. However, for enterprise buildings such as retails and airports, this method of applying the number of turns constraint could improve the map matching accuracy in these areas. We will discuss our method in detail in Section 4.6.3.2.

4.6.2 Smoothing

To reduce the noise level in the location estimation, we smooth the location estimate at the current time t_n with our time-based moving average algorithm. The algorithm uses the window of the latest location estimates and their associated time stamps to adjust the current estimate r_n. A previous estimate that is temporally closer to the current estimate is given a higher weight. The inputs to this algorithm are the size of the smoothing window s ($s \geq 1$), and the observed location estimates until the current time t_n, along with their associated time stamps. The output of this algorithm is the adjusted location estimate $\bar{r}_n$. Algorithm 1 shows the pseudo code of our time-based moving average algorithm.

Algorithm 1 Time-based moving average

Require: $\{(r_1, t_1), ..., (r_n, t_n)\}$, s

Ensure: $\bar{r}_n$

1: $j \leftarrow max(1, n - s + 1)$

2: $total\_weight \leftarrow \sum_{i=j}^{n} t_i - t_j + 1$

3: **for** $i := j$ to n **do**

4: $\quad weight_i \leftarrow \frac{t_i - t_j + 1}{total\_weight}$

5: $\bar{r}_n \leftarrow \sum_{i=j}^{n} weight_i * r_i$

6: **return** $\bar{r}_n$

4.6.3 Modeling

The modeling step models the transitions between path segments when a user walks on a floor by using an HMM. HMMs are general probabilistic models consisting of hidden states and transitions between the states [60]. In the context of the map matching problem, the values of each state variable are possible path segments that

the user can take at a specific time. The actual value of each state is not observed (hidden). Only the user's location estimates are observed. The model assumes that the transition probability to the current state's value (the current path segment) only depends on the previous state's value. The probability of observing the current location estimate only depends on the current state's value.

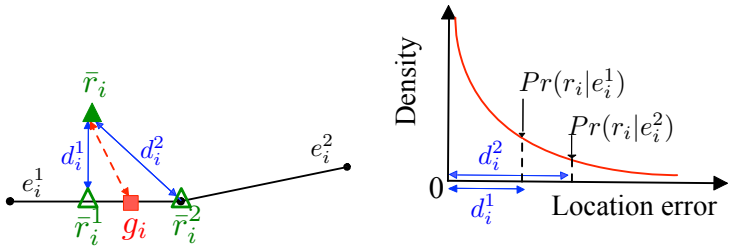
In detail, at any given time step t_i, the modeling step assumes that the correct path segment (edge) belongs to the set E_i consisting of K nearest (candidate) edges to the adjusted location estimate $\bar{r}_i$. The elements of this set are possible values of the current state.

$$E_i = \{e_i^j, j \in 1..K, 0 < K << |E|\}$$

After that, it calculates the probability of emitting the observation $\bar{r}_i$ from each state's value, and the probability of transitioning from a state's value at time step t_{i-1} to a state's value at time step t_i. Below, we describe how this step applies a geometric constraint and topological constraints to calculate the emission probabilities and the transition probabilities.

4.6.3.1 Emission Probability

The modeling step applies a proximity constraint to calculate the probability $Pr(\bar{r}_i|e_i^j)$ of observing the adjusted location estimate $\bar{r}_i$ given the state e_i^j. The probability density is computed based on the Euclidean distance from $\bar{r}_i$ to its nearest point $\bar{r}_i^j$ in e_i^j and the distribution of the location estimation errors estimated from data in the training set. The location error at time step t_i is the Euclidean distance between $\bar{r}_i$ and the corresponding actual location g_i. Section 4.8 shows the location errors follow an exponential distribution. Figure 4.6 shows an example of how this step applies the constraint. The emission probabilities calculated at time step t_i are also normalized



(a) (b)

Figure 4.6: Figure (a) shows the two nearest (candidate) edges e_i^1 and e_i^2 to $\bar{r}_i$ at time t_i. $\bar{r}_i^1$ and $\bar{r}_i^2$ are the two nearest points to $\bar{r}_i$ in e_i^1 and e_i^2, respectively. The dash line represents the distance (the location error at time step t_i) between $\bar{r}_i$ and the corresponding actual location g_i. In Figure (b), given the distribution of the location errors, the probability $Pr(r_i|e_i^1)$ of emitting $\bar{r}_i$ given the edge e_i^1 is computed based on the distance d_i^1.

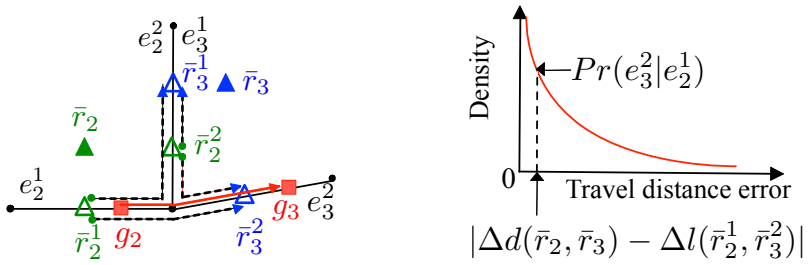
to combine with transition probabilities at this time step.

4.6.3.2 Transition Probability

The modeling step applies topological constraints to calculate the probability $Pr(e_i^k|e_{i-1}^j)$ of transitioning from the state's value e_{i-1}^j at time step t_{i-1} to the state's value e_i^k at time step t_i. Below, we describe how this step applies the topological constraints: travel distance constraint, travel time constraint, and number of turns constraint.

Travel distance constraint We observe that the Euclidean distance $\Delta d(\bar{r}_{i-1}, \bar{r}_i)$ between two consecutive adjusted location estimates $\bar{r}_{i-1}$ and $\bar{r}_i$ is similar to the actual travel distance (length of the path) $\Delta l(g_{i-1}, g_i)$ between the corresponding actual locations g_{i-1} and g_i. In other words, the travel distance error (the absolute difference between these distances) is small. In Section 4.8, we show that the travel distance error follows an exponential distribution.

$$travel\_distance\_error_i = |\Delta d(\bar{r}_{i-1}, \bar{r}_i) - \Delta l(g_{i-1}, g_i)|$$



(a) (b)

Figure 4.7: When a user walks from g_2 to g_3, $\bar{r}_2$ and $\bar{r}_3$ can be aligned on their two nearest edges. Therefore, there are four shortest paths represented by dash lines. The length of the shortest path from $\bar{r}_2^1$ to $\bar{r}_3^2$ approximates $\Delta d(\bar{r}_2, \bar{r}_3)$. Therefore, given e_2^1, transitioning to e_3^2 is more likely than transitioning to e_3^1 ($Pr(e_3^2|e_2^1) > Pr(e_3^1|e_2^1)$). Similarly, given e_2^2, transitioning to e_3^2 is more likely than transitioning to e_3^1.

Based on this observation, the travel distance constraint is that the length of the shortest path $\Delta l(\bar{r}_{i-1}^j, \bar{r}_i^k)$ between two consecutive aligned location estimates on the edges e_{i-1}^j and e_i^k approximates $\Delta d(\bar{r}_{i-1}, \bar{r}_i)$. To apply this constraint, the algorithm computes the transition probability based on the absolute difference between these values and the distribution of the travel distance error. Figure 4.7 shows an example of how the modeling step applies the constraint.

Travel time constraint We observe that the time difference between two consecutive location estimates is similar to the estimated travel time that is the travel distance between the corresponding actual locations divided by the average walking speed of people (obtained from our training set).

$$travel\_time\_error_i = \Delta l(g_{i-1}, g_i)/speed - (t_i - t_{i-1})$$

Based on this observation, the travel time constraint is that the estimated travel time between two consecutive aligned location estimates approximates the difference of the timestamps associated with these estimates. The algorithm applies this constraint in a similar way to the travel distance constraint.

Number of turns constraint As mentioned in Section 4.6.1, we observed that the number of turns that people often make in indoor environments relative to the corresponding walking distance is small, despite environment settings, floor dimensions, and user's motivation for walking. Therefore, we use the normalized frequencies of the number of turns that a participant makes to constrain the number of turns in the shortest paths between two aligned location estimates. Specifically, we sample the number of turns at different walking distances from 1 m to $max\_distance$ m and derive the sample frequencies for each walking distance. Given the length of the shortest path (estimated walking distance) and the number of turns in the path, we look up the normalized frequency of the number of turns. The $max\_distance$ is the maximum length a shortest path can be.

We have described how our algorithm applies the three transition constraints separately. In our work, the algorithm also applies various combinations of these constraints. Assuming the distributions of the travel distance error, the travel time error, and number of turns are independent, the final transition probability from a state's value at time step t_{i-1} to a state's value at time step t_i is the product of the normalized transition probabilities based on these constraints.

4.6.4 Path Segment Selection

Given the HMM from the modeling step, the path segment selection step uses the Viterbi algorithm [60] to infer the optimal value of the states until the current

time step. These values are the most likely path segments that a user can take, given the location estimates. This step outputs the last state's value as the selected path segment for aligning the current estimate.

4.7 Image-based Approach

In this section, we first describe the representation of input data including a floor map and a sequence of location estimates as described in Section 4.5. Then, we describe our approach which is based on a convolutional encoder-decoder architecture [54].

4.7.1 Input Representations

In this section, we describe our approach in representing input data: a floor map and location estimates.

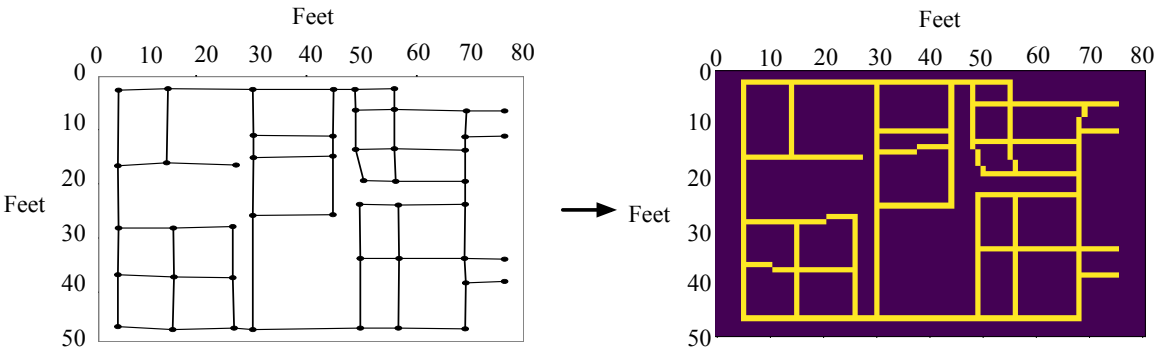


Figure 4.8: Floor map representation: We convert a graph representation of a floor map (left figure) to an image representation of the floor map (right figure) by drawing all path segments on a binary image. The image's pixels whose values are 1 represent the path segments in the floor map. All path segments have the same width, which is 3 feet.

- **Floor map.** We represent a floor map consisting of a set of pre-defined path

segments as a 2D binary image F. The top-left of the image corresponds to the coordinate origin (0,0). Each pixel corresponds to an 1 feet x 1 feet area on a floor. We assume that all path segments have the same width, which is 3 feet. If the area is in a path segment, the pixel value is 1. Otherwise, the pixel value is 0. Figure 4.8 illustrates the floor map representation.

• **Location map.** A location estimate r_i at time step t_i is an estimated coordinate of a device location on a 2D floor. We assume that the location estimation errors follow an isotropic Gaussian distribution. The standard deviation of the distribution σ_l can be estimated from a training data. We represent the location estimate with the Gaussian distribution on a spatial map L_i having the same resolution as the floor map F. Equation 4.1 shows how we compute the value of each pixel p in the image. Figure 4.9 illustrates the image representation L_i of the location estimate r_i.

$$L_i = \exp(\frac{-d(p_{coordinate}, r_i)^2}{2\sigma_l^2})$$
$$\text{where } d(p, r_i) \text{ is the Euclidean distance between each pixel } p \text{ and } r_i \quad (4.1)$$
$$\text{and } \sigma_l \text{ is the standard deviation of location errors}$$

With the input representations, the online map matching problem is defined as follows. Given a floor map F and a sequence of location maps L_1, ..., L_n corresponding to a sequence of location estimates r_1, ..., r_n (r_n is the location estimate at the current time step), the map matching problem is how to constrain the current location estimate r_n on the path taken by a user and close to the user's current location.

• **Spatial map combination.** As a floor map F and a sequence of location maps L_1, ..., L_n are represented as probabilistic maps (whose pixel values range from 0 to 1), they indicate the probabilities of device locations on a floor. Therefore, we can further combine each location map L_i (for $i \in [1, .., n]$) with the floor map F to form a new spatial map I_i. Specifically, we use an element-wise multiplication operation to

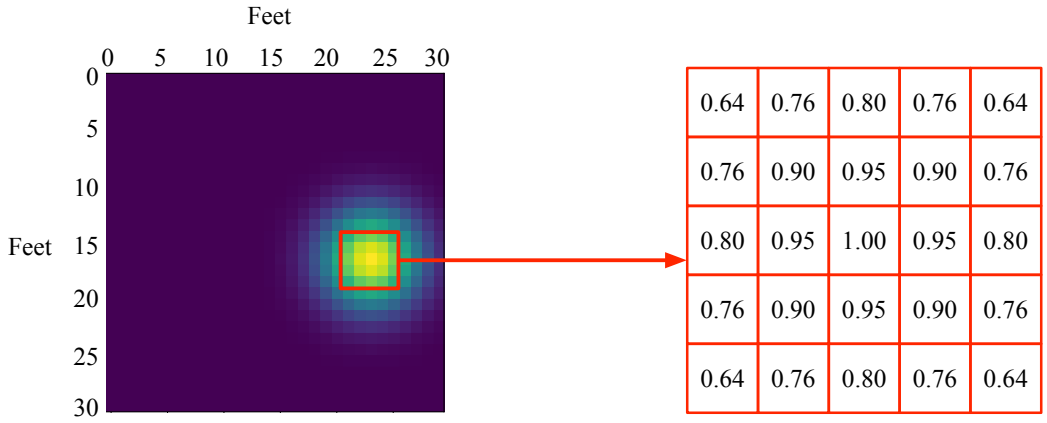


Figure 4.9: Location estimate representation: The left figure shows the representation of a location estimate with a Gaussian noise ($\sigma_l = 3$ feet) on an image, called a *location map*. The right figure shows pixel values within the cropping region (the red square in the left figure). The pixel whose value is 1 corresponds to the location estimate.

combine these spatial maps, as described in Equation 4.2. We illustrate the combined spatial map at a time step in Figure 4.10.

$$I_i = F \odot L_i \tag{4.2}$$

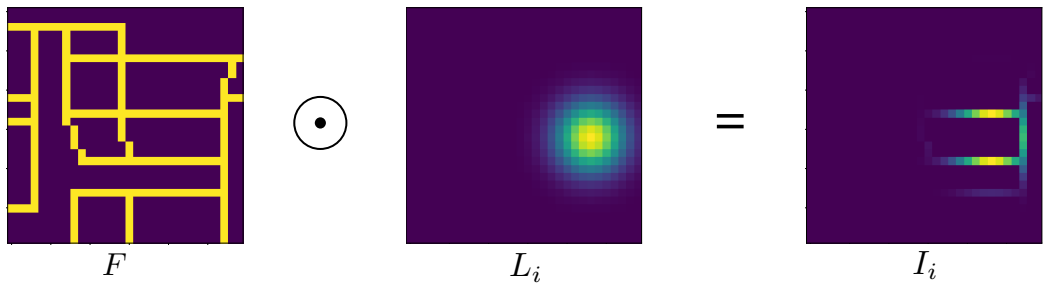


Figure 4.10: Crop regions of a floor map F, a location map L_i at a time step t_i, and the corresponding spatial map I_i generated by using Equation 4.2.

With the combination of the spatial maps at each time step, the online map matching problem is defined as follows. Given a sequence of spatial maps I_1, ...,

I_n corresponding to a sequence of spatial information available at time steps t_1, ..., t_n (t_n is the current time step), the map matching problem is how to constrain the current location estimate r_n on the path taken by a user and close to the user's current location.

Below, we describe the architecture of our deep neural network that exploits the spatial information to address the problem.

4.7.2 Network Architecture

Our 3D convolutional encoder-decoder network takes $I_{n-w+1,...,n}$, a window of w spatial maps $I_{n-w+1}, ..., I_n$ from time steps $t_{n-w+1}, ..., t_w$, and outputs a corresponding decoded spatial maps $I'_{n-w+1,...,n}$. These decoded maps are then converted to the filtered location estimates $r'_{n-w+1}, ..., r'_n$. Finally, we align r'_n on the nearest path segment.

Figure 4.11 shows the network architecture. The encoder consists of 4 components. Each component has a 3D convolution-ReLU layer followed by a 3D Max-Pooling layer. Different from 2D versions of these layers, the 3D versions convolute the information in a spatial domain and then in a temporal domain. This design allows the network to take into account the nature of the input data, which is a sequence of location estimates in the temporal domain. All convolution layers use 3x3x3 (depth x height x width) kernels with stride 1. All Max-Pooling layers use 2x2x2 kernels except for the first two layers in the first two components, which use 1x2x2 kernels. The purpose is to not compress the spatial maps in the temporal domain too early [66]. In other words, we try to retain the spatial information in each time step. The decoder consists of the same number of components as the encoder. Each component has an up-convolution layer followed by a ReLU activation. The final component generates decoded spatial maps $I'_{n-w+1,...,n}$, which are used to compute the corresponding fil-

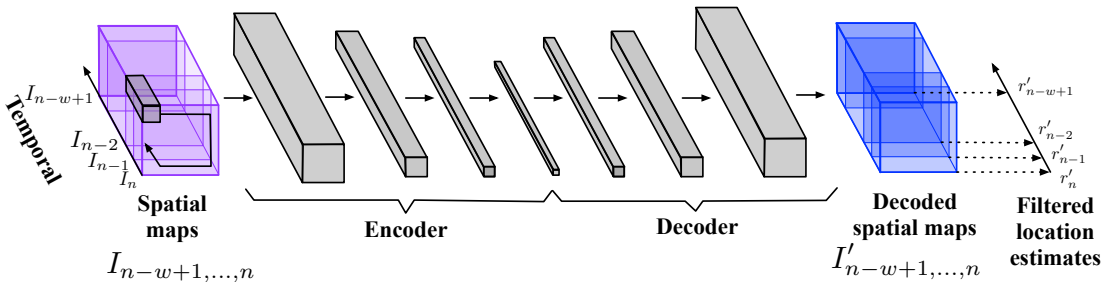


Figure 4.11: Architecture of our 3D convolutional encoder-decoder network. The data input is a stack of w spatial maps $I_{n-w+1}, ..., I_n$. The encoder consists of four components. Each component contains a 3D convolution-ReLU layer followed by a 3D Max-Pooling layer. The first component processes the input and generates a block, which is the input to the next component and so on. The decoder contains four components. Each component performs up-convolution and then ReLU activation. Its final output is a stack of decoded spatial maps having the same dimensions as the input data. The decoded maps $I'_{n-w+1,...,n}$ are used to compute filtered location estimates $r'_{n-w+1}, ..., r'_n$ (Equation 4.3).

tered location estimates $r'_{n-w+1}, ..., r'_n$. As shown in Equation 4.3, we compute the sum of coordinates of all pixels in I'_n weighted by the pixel values to generate r'_n.

$$r'_n = \sum p_{value} * p_{coordinate} \quad \text{for } \forall\ p \in I'_n \tag{4.3}$$

4.7.3 Training

To train the network, we consider different loss functions. In this section, we describe only the loss function that guides the network to exploit a sequence of spatial maps $I_{n-w+1}, ..., I_n$ to improve location estimation process at the current time step t_n. In Equation 4.4, on the left hand side, $L(I'_{n-w+1,...,n}; \theta)$ denotes our loss function in which θ denotes a set of parameters to be optimized. On the right hand side, we

compute the average of location estimation errors. Each of them is the Euclidean distance between a filtered location estimate r'_i and the corresponding actual location g_i for $i \in [n-w+1, n]$.

$$L(I'_{n-w+1,\ldots,n}; \theta) = \frac{1}{w} \sum_{i=n-w+1}^{n} d(r'_i, g_i) \tag{4.4}$$

Training the network requires a large amount of data. Therefore, we simulate trajectories of devices on a floor plan. Then, we use the simulated datasets to train and evaluate our network. In Section 4.8.2 describes our simulation in detail.

4.8 Evaluation

In this section, we describe: (i) the goal of our evaluation, (ii) the datasets that we collect and simulate, (iii) the metrics that we use and define to measure map matching accuracy, (iv) our evaluation methodology, and (v) the experimental results and our analysis.

4.8.1 Goal

The goal of our evaluation is to answer the following questions:

1. For the graph-based approach:
   (a) Do the location estimation errors, the travel distance errors, and the travel time errors follow some probability distribution?
   (b) Which combination of the constraints can be applied to give the best map matching accuracy?
2. For the image-based approach:

(a) Does this approach achieve better map matching accuracy compared to the graph-based approach?

(b) How effectively does the encoder-decoder model exploit the combination of a floor map and location maps?

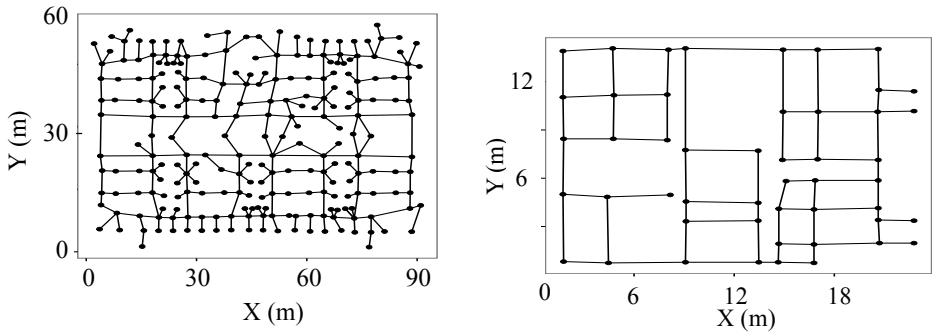
4.8.2 Dataset

In this section, we describe our dataset for training and evaluating the graph-based approach (Section 4.6) and the image-based approach (Section 4.7).

• **Graph-based approach.** We collect data at two different floor plans to train and evaluate our graph-based approach. In our data collection process, we ask a small set of participants *who have not participated in our human mobility study* to perform experiments on two floors having the IIPS deployed by using the mobile application described in Section 4.6.1. Figure 4.12 shows path segments on two floors in two different buildings. During each experiment, we also collect the IIPS-generated location estimates of a participant's mobile device. When the participant finishes the experiment, we interpolate the participant's actual locations at the same time stamps as the location estimates. We also extract the actual path (sequence of path segments) that the participant took during the experiment from the floor map and locations indicated by the participant.

For each experiment, we have a data set containing the actual path and a table whose columns are: $ExperimentID$, $Timestamp$, $RawX$, $RawY$, $ActualX$ and $ActualY$. $RawX$ and $RawY$ are the coordinates of a location estimate. $ActualX$ and $ActualY$ are the the corresponding actual coordinates.

We performed 32 experiments on two different floors that have dense path segments. There are 22 experiments performed by two participants on floor 1 whose area is 93.9 m x 58.5 m. There are 10 experiments performed by one participant on



(a) (b)

Figure 4.12: Floor maps of two different building floors having the IIPS deployed (a) floor 1 (b) floor 2. The dimensions of floor 1 are much larger than floor 2.

floor 2 whose area is 24.7 m x 15.2 m. In each experiment, the participant walks for 5 minutes while holding a mobile device (iPhone5s or Iphone 6 Plus).

• **Image-based approach.** To investigate our encoder-decoder neural network in different environment settings, we build a simulation to generate multiple large datasets having different characteristics to train and evaluate our network. We build a simulation to generate multiple datasets that have the same types of information (*ExperimentID, Timestamp, RawX, RawY, ActualX, ActualY*) as the real datasets that we collected for building and evaluating the graph-based approach. Each dataset consists of multiple trajectories. Each trajectory consists of a sequence of location estimates and corresponding actual locations on a floor plan. We simulate two important characteristics of the real world datasets: sparseness of location estimates and scattering of location estimates. We describe our simulation in detail below. The characteristic that we have not simulated is the lag between location estimates and the corresponding actual locations.

To simulate a trajectory of actual device locations, we first randomly pick two locations in predefined path segments in a floor map F. We then compute the shortest

path between the two locations. To simulate the sparseness of location estimates, we sample the shortest path with a sampling distance s that follows a normal distribution $\mathcal{N}(\mu_s, \sigma_s^2)$. For example, if we sample the shortest path every one meter ($s \sim \mathcal{N}(1, 0)$), we obtain a sequence of locations (a user's trajectory) in the shortest path. To obtain the location estimates corresponding to the locations on the path, we add 2D Gaussian noises into the actual locations. The distribution of the Gaussian noises is sampled on an image (100 x 100) as described in Equation 4.1. We simulate how the location estimates scatter around the corresponding actual locations by using different values of σ_l in the equation. In summary, given a floor map F, there are three parameters $\mu_s, \sigma_s, \sigma_l$ whose values we can change to simulate various datasets denoted as $D_F(\mu_s, \sigma_s, \sigma_l)$.

In our evaluation, we simulate device trajectories in an office space (Figure 4.13). In Section 4.8.4, we describe datasets that we use to evaluate our image-based approach in detail.

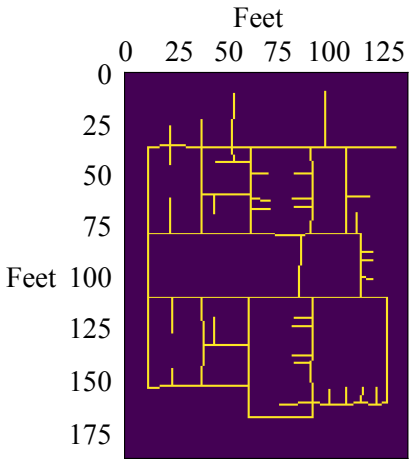


Figure 4.13: A floor map (office space) in our simulation. The map consists of path segments represented as yellow pixels.

4.8.3 Metrics

To evaluate our approaches, we use two main metrics: *location accuracy* and *path-alignment accuracy*.

- *Location accuracy.* Location accuracy represents percentiles of location estimation errors, which are Euclidean distances between aligned location estimates and the corresponding actual locations.
- *Path-alignment accuracy.*

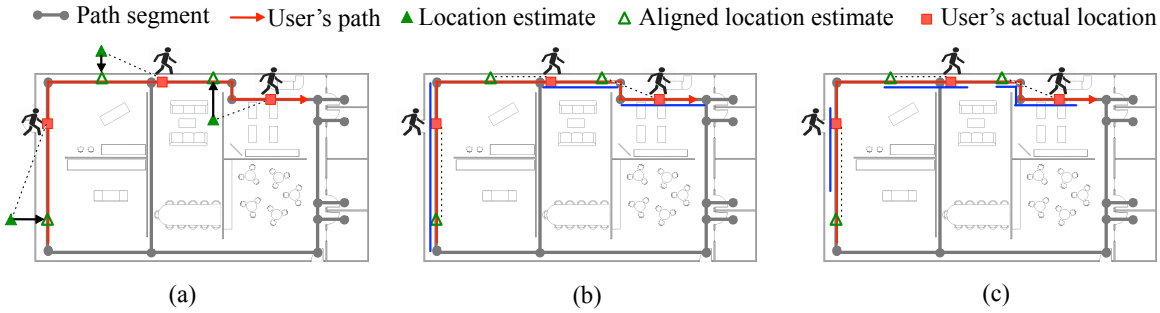


Figure 4.14: Figure (a) shows location estimates often lag the corresponding actual locations, which makes the aligned estimates also lag the corresponding actual locations. Figure (b) shows the path segments (blue lines) containing the actual locations. If aligned location estimates are on the the same path segments as the corresponding actual locations, these aligned estimates are correct map matching results. With this accuracy metric [40, 47], the map matching accuracy is 33%. Figure (c) shows the acceptable paths (blue paths) of actual locations. If aligned estimates are on the acceptable paths of the corresponding actual locations, the aligned estimates are correct map matching results. With our accuracy metric, the map matching accuracy is 67%. Our metric is not affected by the lengths of path segments.

Measuring the path-alignment accuracy of infrastructure-based map matching requires a suitable metric since IIPS-generated location estimates often lag the corre-

sponding actual locations (Figure 4.14a). This happens due to the latency incurred in sampling Wi-Fi signal strengths and in the location estimation process. A prior metric [40, 47] that measures the accuracy as the percentage of the aligned location estimates on the same path segments as the corresponding actual locations is not suitable. This metric is sensitive to the length of a path segment. Figure 4.14b shows an example where map matching accuracy is measured with this metric. Although the first aligned location estimate significantly trails the corresponding actual location, it is considered as a correct map matching result. Other aligned location estimates closer to the corresponding actual locations are not considered as correct map matching results.

In our work, we define a new accuracy metric. First, we define an acceptable path of each actual location. If the location estimate corresponding to the actual location is on an acceptable path, it is considered as a correct map matching result. The idea is that if the aligned location estimate is on the path taken by a user and behind the user's actual location by at most b meter (m), then it is useful for some location-based applications. For example, if a building operator can notify their customers within 3 seconds after they enter a zone, then it is still useful. Based on a human average walking speed in indoor environments [8] which is 1.37 m/s, $b \approx 4$ m. Besides, location-based applications can tolerate a small location estimation error. When a user is stationary, a location estimate is still useful if it is within 1 m from the corresponding actual location. Therefore, if the aligned location estimate is ahead of the actual location by at most 1 m, it is still useful. Figure 4.14c shows examples of acceptable paths. We define our accuracy metric as follows.

Let $Path_n$ be the path taken by a user up to time step t_n. In other words, $Path_n$ is the sequence of path segments (edges) that the user traveled over time until time step t_n. The acceptable path $AcceptPath_i$ containing the actual location g_i ($i \in [1, n]$) is

a set of continuous coordinates $\{c_1, \ldots, c_h\}$ in $Path_n$ ordered in time such that:

- $||c_k - g_i||_2 \approx 0$ for some $k \in [1, h]$ (i.e., $AcceptPath_i$ contains g_i), and
- the length of the path from c_1 to c_k equals b m, and
- the length of the path from c_k to c_h equals 1 m

If the location estimate r_i corresponding to g_i is aligned on $AcceptPath_i$, this is the correct map matching result. In summary, the *path-alignment accuracy* of a map matching algorithm is the percent of location estimates that are aligned on the acceptable paths containing the corresponding actual locations.

- To evaluate the effectiveness of our encoder-decoder model as mentioned in Section 4.8.1, we use the percentage of location estimates that are aligned outside pre-defined paths in a floor map.

4.8.4 Methodology

For each floor data set, we apply the holdout validation method to evaluate our approaches. Below, we describe our methodology to answer our evaluation questions (Section 4.8.1).

- To answer question 1a, we use data in the training sets to calculate (i) location estimation errors, (ii) travel distance errors, and (iii) travel time errors. Then, we fit different well-known distributions (normal distribution, exponential distribution, log-normal distribution etc.) to each of the three empirical data by using a maximum likelihood method [25]. Finally, we evaluate which distribution fits a particular data well by analyzing a q-q plot that shows the quantiles of the empirical data versus the theoretical quantiles of the fitted distribution and a 45-degree reference line [44]. If all points in the plot lie on the reference line, then the distribution is the best fit for the empirical data.

- To answer question 1b, we evaluate alternatives of our map matching algorithm (MM) that apply different combinations of the topological constraints. Table 4.2 shows the map matching approaches we consider. Finally, we measure the map matching accuracy of these approaches for a varying maximum number of candidate path segments considered and with different configurations of our accuracy metric in which the b values are: 4 m, 5 m, and 6 m. These values are based on the expected latency in location-based services which are: 3 seconds, 4 seconds, and 5 seconds (respectively). Based on our preliminary user study, location-based services often expect about three-second latency.

Constraint \ Approach	MM-D	MM-T	MM-N	MM-DT	MM-TN	MM-DN	MM-DNT
Travel time		✓		✓	✓		✓
Travel distance	✓			✓		✓	✓
Number of turns			✓		✓	✓	✓

Table 4.2: Map matching approaches that apply different combinations of topological constraints

- To answer questions 2a and 2b, we use our simulation (described in Section 4.8.2) to generate three groups of datasets: $D_F(\mu_s = 3, \sigma_s = 0, \sigma_l)$, $D_F(\mu_s = 15, \sigma_s = 0, \sigma_l)$, and $D_F(\mu_s = 15, \sigma_s = 13, \sigma_l)$. As described in Section 4.8.2, μ_s and σ_s represent the mean and standard deviation of consecutive locations, respectively. σ_l represents the standard deviation of the location estimation errors. To generate datasets in each group, we use different values of σ_l: 3, 6, 9, 12, and 15. All parameter values are in feet.

  - To answer question 2a, we compare the location accuracy and path-alignment accuracy between the image-based approach and the graph-based approach

(MM-DN).

– To answer question 2b, we compare the percentage of location estimates output by the image-based approach that outside the pre-defined path segments in two cases. First, the image-based approach only uses location maps. Second, the image-based approach uses combinations of location maps and the floor map.

4.8.5 Results and Analysis

Below, we describe our results and analysis of the graph-based approach and the image-based approach.

• **Graph-based approach.** Below, we answer the first two questions as described in Section 4.8.1.

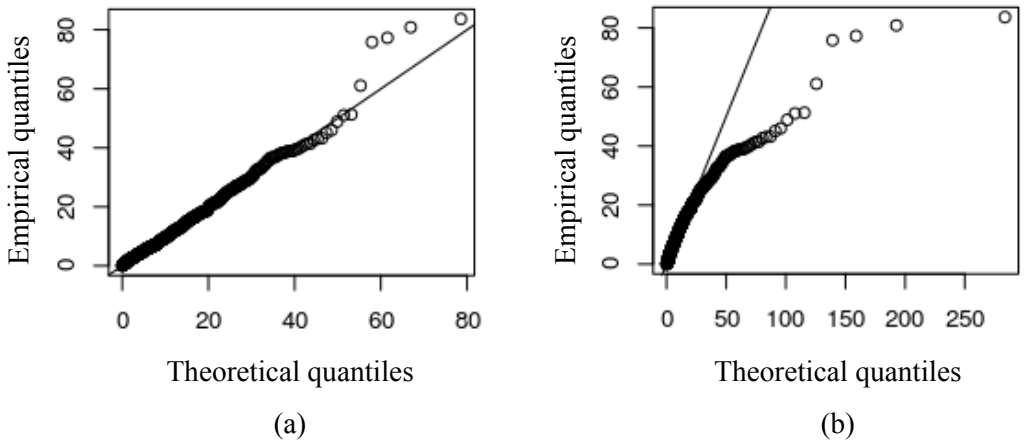
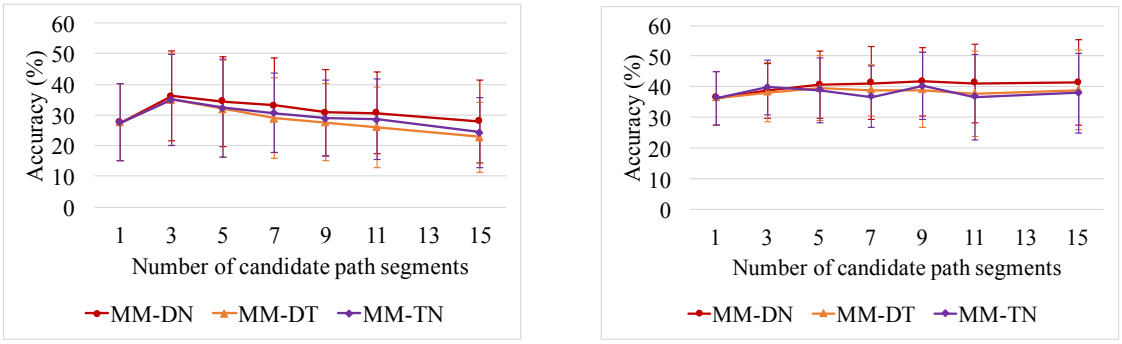


Figure 4.15: q-q plots of the quantiles of travel distance errors (empirical data) in floor 1 versus (a) the quantiles of a fitted exponential distribution and (b) the quantiles of a fitted log-normal distribution. The travel distance errors follow the exponential distribution better.

Distributions. Figure 4.15 shows the quantiles of the travel distance errors (empirical data) on floor 1 versus (a) the theoretical quantiles of the exponential distribution

and (b) the theoretical quantiles of the log-normal distribution. Since most of the points in the plot (a) lie in the reference line, the exponential distribution fits the empirical data better than the log-normal distribution. For floor 2, we have similar q-q plots. We also find that the location estimation errors and travel time errors follow an exponential distribution and a normal distribution quite well, respectively.



(a) (b)

Figure 4.16: Path-alignment accuracy of map matching approaches with the acceptable paths having $b = 4$ m versus the number of candidate path segments on (a) floor map 1 (Figure 4.12a) and (b) floor map 2 (Figure 4.12b). MM-DN that applies the travel distance constraint and number of turns constraint outperforms other approaches.

Topological constraints. First, we consider the map matching accuracy when the b value of the acceptable paths equals 4 m. Figure 4.16 shows the accuracy of different map matching approaches (MM-DN, MM-DT, and MM-TN) described in Table 4.2 versus the maximum number of candidate (nearest) path segments that the algorithm considers. We did not show the accuracy of MM-D, MM-T and MM-N because we find that they have lower accuracy than the approaches shown in the figure. MM-DNT has lower accuracy than MM-DN. From the figure, the approaches that apply the number of turns constraint have higher accuracy on different floor maps. MM-DN

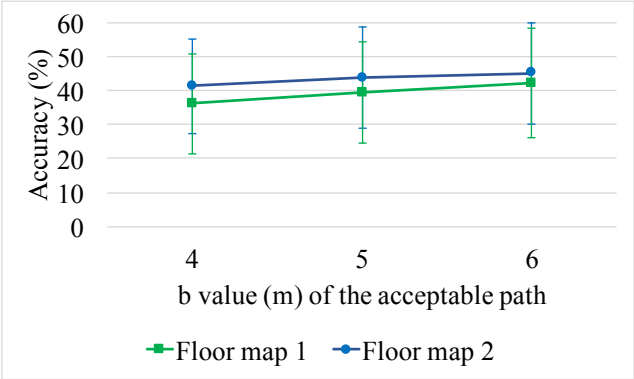


Figure 4.17: Path-alignment accuracy of MM-DN versus the b value of the acceptable path.

has the best accuracy which indicates that applying the travel distance constraint and the number of turns constraint gives the best map matching accuracy regardless of the number of candidate path segments considered.

When the b value of the acceptable path increases from 4 m to 6 m, the accuracy of our map matching algorithm increases. Figure 4.17 shows the accuracy of MM-DN increases as the b value increases. This implied that our map matching algorithm aligns location estimates on the path segments taken by mobile users but the alignments often lag the corresponding actual locations. This happens because the location estimates often lag the corresponding actual locations.

• **Image-based approach.** Below, we first compare the map matching results of image-based approach the graph-based approach. We then discuss the effectiveness of our image-based approach in exploiting a floor map.

Image-based approach versus graph-based approach. Figure 4.18a and 4.18b shows the path-alignment accuracy and the location accuracy of both approaches with the simulated dataset $D_F(\mu_s = 15, \sigma_s = 13, \sigma_l = 15)$, respectively. Figure 4.18a shows that the graph-based approach achieves a better path-alignment accuracy compared to the image-based approach. This result indicates that the graph-based approach

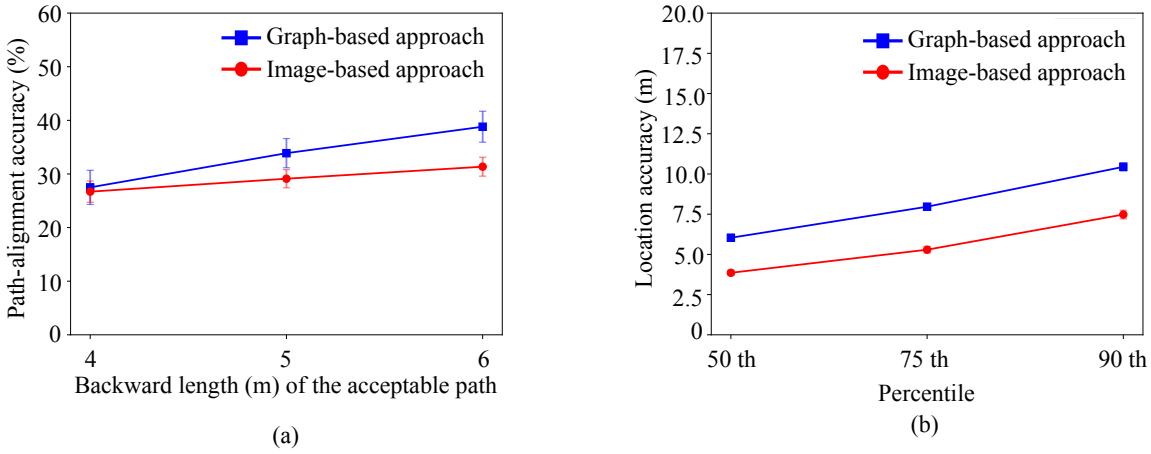


Figure 4.18: (a) Path-alignment accuracy and (b) location accuracy of both the graph-based approach and the image-based approach with the simulated dataset $D_F(\mu_s = 15, \sigma_s = 13, \sigma_l = 15)$

can select path segments more correctly for aligning location estimates. However, the image-based approach achieves a much better location accuracy as shown in Figure 4.18b. This result indicates that the image-based approach was guided to improve the location accuracy and less constrained on finding correct path segments. We achieved similar results for the other simulated datasets.

Compared to the image-based framework, the graph-based approach achieves better path-alignment accuracy by applying the combination of different geometric constraints and topological constraints to select the best path segment for aligning location estimates.

Effectiveness of our image-based approach in exploiting a floor map. We further analyze the effectiveness of our encoder-decoder neural network in exploiting a floor map. Figure 4.19 shows the percent of location estimates output by the network that are outside of the path segments in the floor map in two cases. First, the input to the neural network contains combinations of the floor map and location maps. Second, the

input contains only location maps. When using the floor map, the percentage of the location estimates outside the path segments is reduced significantly. We observed similar results for other simulated datasets. The result indicates the effectiveness of using the location map. Specifically, this effectiveness of the model is mainly attributed to the operator used to combine the floor map and location maps (Section 4.7.1).

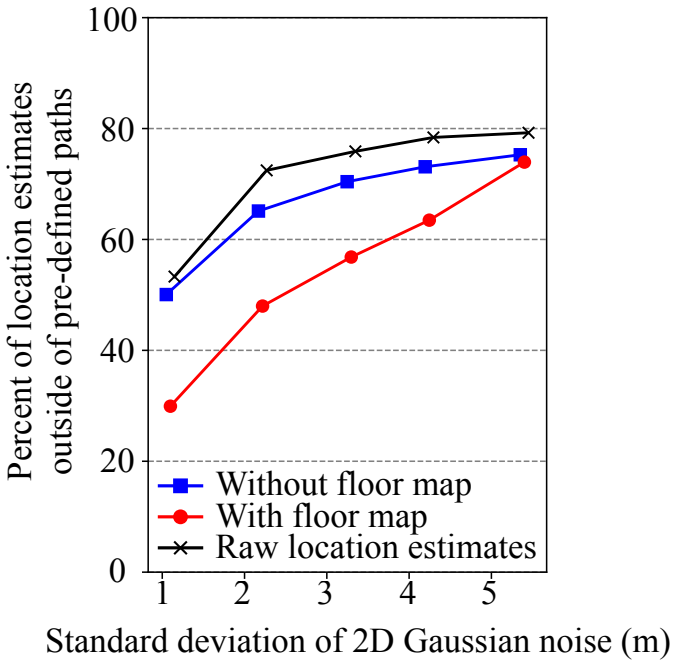


Figure 4.19: Percent of location estimates that are outside of the pre-defined path segments in a floor map in three cases. First, the location estimates are raw without applying any map matching. Second, location estimates are output from the image-based approach using only location maps. Third, location estimates are output from the image-base approach using combinations of the floor map and location maps.

In summary, the image-based approach can potentially exploit the floor map to improve location accuracy and constraint location estimates on the path-segments. The location estimates output from the neural networks are not strictly constrained on

path-segments because we only minimize the average location errors during training the network (as described in Section 4.7.3). In the future work, we will investigate different loss functions to improve path-alignment accuracy of the neural network.

4.9 Discussion

In this section, we compare and contrast the graph-based approach and the image-based approach in Table 4.3 and describe directions for future work to build on these two approaches.

Graph-based approach.

• The contribution of our work is to show that the number of turns constraint and the travel distance constraint can be applied to achieve a better map matching accuracy with data from the IIPS. Future work could consider how to combine these constraints more effectively [50].

• Better models such as HMM of higher orders could be used to consider the constraints not only between two consecutive location estimates but also between several consecutive location estimates.

• Better path segment selection method [40, 71] can be applied to improve the map matching accuracy.

Image-based approach.

• Different loss functions can be considered and combined to guide the encoder-decoder network to learn constraints to improve path-alignment accuracy.

• Simulated training sets and real training sets can be combined to reduce efforts in collecting data for training the network. For example, one could use the simulated data to first guide the network to learn how to encode a floor map and then use the real data to continue training the network.

• Different combinations of hyper-parameter values can be further investigated to

	Graph-based approach	**Image-based approach**
Input representations	A graph representing a floor map and a sequence of location estimates	A sequence of spatial maps, each of which is a combination of two images: a floor map and a location map
Model	The HMM model of order one that only considers the constraints based on consecutive location estimates	The 3D convolutional encoder-decoder network that can consider the spatial information of a sequence of spatial maps
Constraints	Requiring explicit geometric constraints and topological constraints based on heuristics	Implicit constraints can be learned by the model from the training data. However, the current network does not learn the constraints effectively.
Training dataset	Requiring a small training data (collected in several hours)	Requiring a much larger training data
Location accuracy	Not reducing location error	Reducing location error significantly
Path-alignment accuracy	Achieving a better path-alignment accuracy compared to the nearest-path selection approach	Achieving a lower path-alignment accuracy compared to the graph-based approach

Table 4.3: Summary and comparison of our map matching approaches: the graph-based approach and the image-based approach

help the network deal with sparseness of location estimates. In the current version of the network, all of the kernel sizes of the 3D convolutional layers and Max-Pooling are small and the same.

4.10 Conclusion

In summary, we have shown that our map matching approaches can exploit a floor map and a sequence of noisy, temporally sparse location estimates to improve tracking accuracy in term of path-alignment accuracy or location accuracy. In our graph-based approach, we study the constraints that could be applied to achieve the best map matching accuracy with data from the IIPS. From our preliminary study of the number of turns that people often make in indoor environments and the analysis of the trajectories of location estimates, we showed that the point-to-curve proximity constraint, travel distance constraint, and number of turns constraint can be applied for robust map matching on IIPS data. In our image-based approach, we have shown that it is possible that the deep neural network can learn features directly from training data to improve location accuracy. By using images to represent spatial data, we can easily exploit other contexts besides the floor map such as a Wi-Fi coverage map representing how good Wi-Fi signals across a floor, or an obstacle map representing obstacles on a floor.

5 Conclusion

To conclude this dissertation, we summarize our contributions and give directions for future work.

5.1 Research Contributions

In this dissertation, we focused on addressing the problem of scalable and accurate localization and tracking with the IIPS. Addressing the problem is extremely challenging because of the limited computational resources of the IIPS and noisy, temporally sparse Wi-Fi measurements. We explored the notion of device context present in indoor environments. The key idea in our solution is to detect and exploit contexts of devices which help the IIPS scale and track devices efficiently and accurately. In particular, we showed that device motion context and floor map context are two critical contexts that can be leveraged to address the problem. Below, we summarise our two main contributions towards enabling context-aware IIPS.

First, device motion context (stationary or moving) can help the IIPS prioritize expensive location computation as well as combining past, temporally sparse measurements to improve localization accuracy. In our work, we showed that device motion can be detected accurately in real time by using only noisy, temporally sparse, and partial Wi-Fi measurements at the infrastructure-side. The key idea in our approach is to exploit the relationships between the device motion and temporal-spatial fea-

tures of measurements at multiple access points (APs). We proposed two different models: feature-based models and deep learning end-to-end models. We focused on simplicity, accuracy, and generalizability in designing these models. Our models consist of three steps: a feature extraction step for extracting temporal features from measurements at each AP, a feature aggregation step for aggregating the temporal features at multiple APs, and a modeling step for learning a relationship between the temporal-spatial features with device motion. Compared to the feature-based models, the end-to-end models can achieve better motion classification accuracy. However, they are less generalized across environment settings, having a higher computational cost, and requiring a larger training dataset.

The second important context is a floor map consisting of path segments that a device user can take. Therefore, the floor map gives critical spatial information about the device location. However, performing map matching, i.e., leveraging the floor map to improve tracking the device, is challenging due to noisy, temporally sparse location estimates and high density of the path segments. In our work, we proposed and evaluated two map matching approaches: a graph-based approach and an image-based approach. For the graph-based approach, the floor map is represented as an undirected graph. We developed a Hidden Markov Model (HMM) to encode a geometric constraint and topological constraints to select path segments for aligning location estimates. In particular, we proposed and encoded a constraint on how frequently a person makes a turn into the HMM to improve path-alignment accuracy. For the image-based approach, the key idea is to represent both the floor map and location estimates as spatial maps (images). Given these spatial maps, we trained a 3D convolutional encoder-decoder network to leverage the input data to improve tracking accuracy. Our evaluation with a large simulated dataset shows that this approach can improve location accuracy significantly. However, the path-alignment

accuracy is smaller than the graph-based approach.

5.2 Future Directions

We have summarized our two main contributions towards enabling the context-aware IIPS that can track many devices simultaneously and accurately. We believe that there are many other kinds of contextual information that we can exploit to improve the IIPS performance. Below, we describe several other contexts that provide directions for future work as well as a framework for exploiting these contexts.

There are several contexts that we can exploit to improve the performance of the IIPS in terms of scalability and tracking accuracy. First, similar to the floor map, an *obstacle map* consisting of obstacles such as walls and furniture can be exploited to constraint location estimates. The map can be extracted from a floor plan available in several image formats at an infrastructure side. Among these formats, a Computer-Aided Design (CAD) image contains precision drawings required to construct the floor. Therefore, fixed obstacles such as walls can be extracted accurately. Second, the *vicinal devices* context is about devices (peers) that are spatially close to a device in a period. Similar to exploiting device motion context, we can exploit this context to improving the scalability of the IIPS by localizing groups of devices instead of individual devices or improving tracking accuracy by combining measurements of Wi-Fi signals from peer devices [39]. Third, the *convex-hull map* of APs is an important context for a device location. At a location, a device can be inside or outside of a convex hull of APs which receive signals from the device. When the device is inside the convex hull, range-based localization methods often yield better location estimation accuracy [49]. Multiple localization methods take into account this context to improve location accuracy [49]. Besides these contexts, we believe there are other contexts such as Wi-Fi coverage map that can be exploited to improve the performance of the IIPS.

How we leverage these contexts is an important factor in improving localization and tracking accuracy of the IIPS. We observed that multiple contexts such as the floor map, the obstacle map, and the convex hull can be represented as 2D images. Besides, a coordinate of a location estimate or the likelihoods of the device location on a 2D floor can also be represented on 2D images. We plan to extend our image-based framework for map matching to exploit all these contexts. We believe that the framework can enable exploiting these contexts simultaneously and effectively to improve localization and tracking accuracy of the IIPS. We envision this framework can also be applied to other indoor positioning systems.

In conclusion, we believe that leveraging device contexts is one of the key solutions to improving scalability as well as localization and tracking accuracy of the IIPS. Besides, contexts such as the device motion and the vicinal devices can provide useful information for analyzing behaviors of device users.

Bibliography

[1] Al-Sadoon, M.A., Ali, N.T., Dama, Y., Zuid, A., Jones, S.M., Abd-Alhameed, R.A., Noras, J.M.: A new low complexity angle of arrival algorithm for 1d and 2d direction estimation in mimo smart antenna systems. Sensors 17(11), 2631 (2017)

[2] Aly, H., Youssef, M.: semmatch: Road semantics-based accurate map matching for challenging positioning data. In: Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems. pp. 5:1–5:10. SIGSPATIAL '15, ACM, New York, NY, USA (2015), http://doi.acm.org/10.1145/2820783.2820824

[3] Aruba: Integrated Aruba solution supports over 100 healthcare applications, and establishes platform for innovation. https://www.arubanetworks.com/en-gb/resources/jihlava-hospital/ (April 2019)

[4] Bahl, P., Padmanabhan, V.N.: Radar: an in-building rf-based user location and tracking system. In: Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064). vol. 2, pp. 775–784 vol.2 (2000)

[5] Basiri, A., Lohan, E.S., Moore, T., Winstanley, A., Peltola, P., Hill, C., Amirian, P., e Silva, P.F.: Indoor location based services challenges, requirements and usability of current solutions. Computer Science Review 24, 1 – 12 (2017), http://www.sciencedirect.com/science/article/pii/S1574013716301782

[6] Bernstein, D., Kornhauser, A.: An introduction to map matching for personal navigation assistants. Tech. rep., New Jersey TIDE Center, Newark, NJ, USA (1996)

[7] Brakatsoulas, S., Pfoser, D., Salas, R., Wenk, C.: On map-matching vehicle tracking data. In: Proc. of the 31st Int. Conf. on Very Large Data Bases. pp. 853–864. VLDB Endowment, Trondheim, Norway (2005)

[8] Brogan, D.C., Johnson, N.L.: Realistic human walking paths. In: Computer Animation and Social Agents, 2003. 16th International Conference on. pp. 94–101. IEEE (2003)

[9] Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond euclidean data. IEEE Signal Processing Magazine 34(4), 18–42 (2017)

[10] Chapre, Y., Ignjatovic, A., Seneviratne, A., Jha, S.: Csi-mimo: Indoor wi-fi fingerprinting system. In: 39th Annual IEEE Conference on Local Computer Networks. pp. 202–209 (Sept 2014)

[11] Cherian, J., Luo, J., Guo, H., Ho, S., Wisbrun, R.: Parkgauge: Gauging the occupancy of parking garages with crowdsensed parking characteristics. In: 2016 17th IEEE International Conference on Mobile Data Management (MDM). vol. 1, pp. 92–101 (June 2016)

[12] Chiheb Trabelsi, Olexa Bilaniuk, Y.Z.D.S.S.S.J.F.S.S.M.N.R.Y.B.C.J.P.: Deep complex networks. arXiv preprint arXiv:1705.09792 (2017)

[13] Chintalapudi, K., Padmanabha Iyer, A., Padmanabhan, V.N.: Indoor localization without the pain. In: Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking. pp. 173–184. MobiCom '10, ACM, New York, NY, USA (2010), http://doi.acm.org.proxy.lib.pdx.edu/10.1145/1859995.1860016

[14] Cisco: Wireless LAN. https://www.cisco.com/c/m/en_ae/solutions/wireless-lan.html (2009)

[15] Cisco: Cisco Aironet Access Point Wireless Security Module (WSM). https://www.cisco.com/c/en/us/products/collateral/interfaces-modules/aironet-access-point-module-wireless-security-spectrum-intelligence/white_paper_c11-723471.html (October 2014)

[16] Cisco: Cisco Connected Mobile Experiences (CMX) CVD. https://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Borderless_Networks/Unified_Access/CMX/CMX_DeploymentModels.html (September 2014)

[17] Cisco: CMX Deployment Models. https://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Borderless_Networks/Unified_Access/CMX/CMX_DeploymentModels.html (December 2015)

[18] Cisco: CMX FastLocate Deployment Guide MSE Release 8.0. https://www.cisco.com/c/en/us/td/docs/wireless/controller/technotes/8-0/CMX_FastLocate_DG/b_CMX-FastLocate-DG.html (January 2015)

[19] Cisco: Cisco Wireless Controllers. https://www.cisco.com/c/dam/en/us/products/collateral/interfaces-modules/services-modules/at_a_glance_c45-652653.pdf (2016)

[20] Cisco: Hyperlocation: Best Practices and Troubleshooting Guide. https://goo.gl/J127P5 (December 2016)

[21] Cisco: Cisco CMX Configuration Guide. https://www.cisco.com/c/en/us/td/docs/wireless/mse/10-4/cmx_config/b_cg_cmx104/the_cisco_cmx_detect_and_locate_service.html (August 2018)

[22] Cisco: Cisco Connected Mobile Experiences Data Sheet. https://www.cisco.com/c/en/us/solutions/collateral/enterprise-networks/connected-mobile-experiences/datasheet-c78-734648.html (April 2018)

[23] Cisco: Cisco Hyperlocation Module with Advanced Security Data Sheet. https://www.cisco.com/c/en/us/products/collateral/interfaces-modules/aironet-hyperlocation-module-advanced-security/datasheet-c78-734901.html (December 2018)

[24] Cisco: Cisco Aironet 4800 Access Points Data Sheet. https://www.cisco.com/c/en/us/products/collateral/wireless/aironet-4800-access-point/nb-09-air-4800-acces-ds-cte.html (February 2019)

[25] Delignette-Muller, M.L., Dutang, C., et al.: fitdistrplus: An R package for fitting distributions. J. of Statistical Software 64(4), 1–34 (2015)

[26] García-Laencina, P.J., Sancho-Gómez, J.L., Figueiras-Vidal, A.R.: Pattern classification with missing data: a review. Neural Computing and Applications 19(2), 263–282 (2010)

[27] Hart, B.D., Stager, P.J., Pandey, S., Kloper, D., Lyons, D., Silverman, M.A.: Angle of arrival location sensing with antenna array (Nov 21 2017), US Patent 9,823,330

[28] Hemminki, S., Nurmi, P., Tarkoma, S.: Accelerometer-based transportation mode detection on smartphones. In: Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems. SenSys '13 (2013)

[29] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. 9(8), 1735–1780 (Nov 1997), http://dx.doi.org/10.1162/neco.1997.9.8.1735

[30] Hsu, C.Y., Liu, Y., Kabelac, Z., Hristov, R., Katabi, D., Liu, C.: Extracting gait velocity and stride length from surrounding radio signals. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. pp. 2116–2126. CHI '17, ACM, New York, NY, USA (2017), http://doi.acm.org/10.1145/3025453.3025937

[31] Joshi, K., Bharadia, D., Kotaru, M., Katti, S.: Wideo: Fine-grained device-free motion tracing using RF backscatter. In: 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15). pp. 189–204. USENIX Association, Oakland, CA (2015), https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/joshi

[32] Kavitha Muthukrishnan, K., van der Zwaag, B., Havinga, P.: Inferring motion and location using WLAN RSSI, pp. 163–182. Lecture Notes in Computer Science, Springer Verlag (9 2009)

[33] Kotaru, M., Joshi, K., Bharadia, D., Katti, S.: Spotfi: Decimeter level localization using wifi. In: Proceedings of the 2015 ACM Conference on Special Interest

Group on Data Communication. pp. 269–282. SIGCOMM '15, ACM, New York, NY, USA (2015), http://doi.acm.org/10.1145/2785956.2787487

[34] Krumm, J., Horvitz, E.: Locadio: inferring motion and location from wi-fi signal strengths. In: The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. pp. 4–13 (Aug 2004)

[35] Krumm, J., Horvitz, E., Letchner, J.: Map matching with travel time constraints. Tech. Rep. 2007-01-1102, SAE Technical Paper, Detroit, MI, USA (2007)

[36] Levey, A., Lindenbaum, M.: Sequential karhunen-loeve basis extraction and its application to images. IEEE Transactions on Image processing 9(8), 1371–1374 (2000)

[37] Office employees should be on feet for four hours of working day, study says. https://goo.gl/HL2Xf8 (June 2015)

[38] Lipton, Z.C., Kale, D.C., Wetzel, R.: Modeling missing data in clinical time series with rnns. Machine Learning for Healthcare (2016)

[39] Liu, H., Yang, J., Sidhom, S., Wang, Y., Chen, Y., Ye, F.: Accurate wifi based localization for smartphones using peer assistance. IEEE Transactions on Mobile Computing 13(10), 2199–2214 (Oct 2014)

[40] Lou, Y., Zhang, C., Zheng, Y., Xie, X., Wang, W., Huang, Y.: Map-matching for low-sampling-rate GPS trajectories. In: Proc. of the 17th ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems. pp. 352–361. ACM, Seattle, WA, USA (2009)

[41] Meraki, C.: Proximity marketing. https://create.meraki.io/guides/proximity-marketing/ (April 2019)

[42] Mohamed, R., Aly, H., Youssef, M.: Accurate and efficient map matching for challenging environments. In: Proceedings of the 22Nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. pp. 401–404. SIGSPATIAL '14, ACM, New York, NY, USA (2014), http://doi.acm.org/10.1145/2666310.2666429

[43] Muthukrishnan, K., Lijding, M., Meratnia, N., Havinga, P.: Sensing motion using spectral and spatial analysis of wlan rssi. Smart Sensing and Context pp. 62–76 (2007)

[44] National Institute of Standards and Technology: Quantile-quantile plot. http://www.itl.nist.gov/div898/handbook/eda/ section3/qqplot.htm (2013)

[45] Union Agency for Network, E., Information Security: Passive WIFI Surveillance and Access Point Hijacking. https://www.enisa.europa.eu/publications/info-notes/passive-wifi-surveillance-and-access-point-hijacking (2015)

[46] School News, A.: UW's AccessMap makes navigating Seattle safer and more accessible for all. https://news.cs.washington.edu/2017/02/02/uws-accessmap-makes-navigating-seattle-safer-and-more-accessible-for-all (February 2017)

[47] Newson, P., Krumm, J.: Hidden Markov map matching through noise and sparseness. In: Proc. of the 17th ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems. pp. 336–343. ACM, Seattle, WA, USA (2009)

[48] Nick, T., Coersmeier, E., Geldmacher, J., Goetze, J.: Classifying means of transportation using mobile sensor data. In: Neural Networks (IJCNN), The 2010 International Joint Conference on. pp. 1–6. IEEE (2010)

[49] O'Lone, C.E., Buehrer, R.M.: An analysis of the convex hull's impact on localization performance. In: 2016 IEEE/ION Position, Location and Navigation Symposium (PLANS). pp. 519–526 (April 2016)

[50] Osogami, T., Raymond, R.: Map matching with inverse reinforcement learning. In: IJCAI (2013)

[51] Pandey, S., Rong, P.: Determining location via wireless access points (Jun 27 2017), US Patent 9,693,194

[52] Quddus, M.A., Ochieng, W.Y., Zhao, L., Noland, R.B.: A general map matching algorithm for transport telematics applications. J. GPS Solutions 7(3), 157–167 (2003)

[53] Rana, R.K., Chou, C.T., Kanhere, S.S., Bulusu, N., Hu, W.: Ear-phone: An end-to-end participatory urban noise mapping system. In: Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks. pp. 105–116. IPSN '10, ACM, New York, NY, USA (2010), http://doi.acm.org/10.1145/1791212.1791226

[54] Ranzato, M., Huang, F.J., Boureau, Y., LeCun, Y.: Unsupervised learning of invariant feature hierarchies with applications to object recognition. In: 2007 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–8 (June 2007)

[55] Ross, D.A., Lim, J., Lin, R.S., Yang, M.H.: Incremental learning for robust visual tracking. International journal of computer vision 77(1-3), 125–141 (2008)

[56] Roy, R., Kailath, T.: Esprit-estimation of signal parameters via rotational invariance techniques. IEEE Transactions on acoustics, speech, and signal processing 37(7), 984–995 (1989)

[57] Schmidt, R.: Multiple emitter location and signal parameter estimation. IEEE Transactions on Antennas and Propagation 34(3), 276–280 (Mar 1986)

[58] Scroxton, A.: Birmingham Airport uses sensor tracking to reduce queue times. https://www.computerweekly.com/news/450414603/Birmingham-Airport-uses-sensor-tracking-to-reduce-queue-times (March 2017)

[59] Seitz, J., Jahn, J., Boronat, J.G., Vaupel, T., Meyer, S., Thielecke, J.: A Hidden Markov Model for urban navigation based on fingerprinting and pedestrian dead reckoning. In: Proc. of the 13th Int. Conf. on Information Fusion. pp. 1–8. IEEE, Edinburgh, UK (2010)

[60] Srámek, R., Brejová, B., Vinar, T.: On-line Viterbi algorithm and its relationship to random walks. CoRR abs/0704.0062 (2007), http://arxiv.org/abs/0704.0062

[61] Sun, L., Sen, S., Koutsonikolas, D.: Bringing mobility-awareness to wlans using phy layer information. In: Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies. pp. 53–66. CoNEXT '14, ACM, New York, NY, USA (2014), http://doi.acm.org/10.1145/2674005.2675017

[62] Tensorflow. https://www.tensorflow.org/ (December 2017)

[63] Dense layer. https://www.tensorflow.org/api_docs/python/tf/layers/dense (December 2017)

[64] Gradient descent optimizer. https://goo.gl/6p6cH6 (December 2017)

[65] Thrun, S.: Particle filters in robotics. In: in Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI (2002)

[66] Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3d convolutional networks. In: Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV). pp. 4489–4497. ICCV '15, IEEE Computer Society, Washington, DC, USA (2015), http://dx.doi.org/10.1109/ICCV.2015.510

[67] Tran, H., Mukherji, A., Bulusu, N., Pandey, S., Zhang, X.: Improving infrastructure-based indoor positioning systems with device motion detection. In: 2019 IEEE International Conference on Pervasive Computing and Communications (PerCom). IEEE (2019)

[68] Tran, H., Pandey, S., Bhattacharyya, A., Natarajan, P., Bulusu, N.: Glimpsing into the future: An accurate wifi-based client count prediction algorithm. 17th International Workshop on Mobile Computing Systems and Applications (2016), http://www.hotmobile.org/2016/index.php?id=demo

[69] Tran, H., Pandey, S., Bulusu, N.: Online map matching for passive indoor positioning systems. In: Proceedings of the 8th ACM SIGSPATIAL Workshop on GeoStreaming. pp. 1–10. IWGS'17, ACM, New York, NY, USA (2017), http://doi.acm.org/10.1145/3148160.3148161

[70] Tran, H., Pandey, S., Bulusu, N.: Poster: Online map matching for passive indoor positioning systems. In: Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services. pp. 175–175. Mo-

biSys '17, ACM, New York, NY, USA (2017), http://doi.acm.org/10.1145/3081333.3089319

[71] Wang, G., Zimmermann, R.: Eddy: An error-bounded delay-bounded real-time map matching algorithm using HMM and online Viterbi decoder. In: Proc. of the 22nd ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems. pp. 33–42. ACM, New York, NY, USA (2014)

[72] Wang, H., Zhang, D., Ma, J., Wang, Y., Wang, Y., Wu, D., Gu, T., Xie, B.: Human respiration detection with commodity wifi devices: Do user location and body orientation matter? In: Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing. pp. 25–36. UbiComp '16, ACM, New York, NY, USA (2016), http://doi.acm.org/10.1145/2971648.2971744

[73] Wang, X., Gao, L., Mao, S.: Phasefi: Phase fingerprinting for indoor localization with a deep learning approach. In: 2015 IEEE Global Communications Conference (GLOBECOM). pp. 1–6 (Dec 2015)

[74] Welch, G., Bishop, G., et al.: An introduction to the Kalman filter (1995)

[75] White, C.E., Bernstein, D., Kornhauser, A.L.: Some map matching algorithms for personal navigation assistants. J. Transportation Research Part C: Emerging Technologies 8(1), 91–108 (2000)

[76] Wu, C., Yang, Z., Zhou, Z., Liu, X., Liu, Y., Cao, J.: Non-invasive detection of moving and stationary human with wifi. IEEE Journal on Selected Areas in Communications 33(11), 2329–2342 (Nov 2015)

[77] Xiao, J., Wu, K., Yi, Y., Wang, L., Ni, L.M.: Fimd: Fine-grained device-free motion detection. In: 2012 IEEE 18th International Conference on Parallel and Distributed Systems. pp. 229–235 (Dec 2012)

[78] Xiong, J., Jamieson, K.: Arraytrack: A fine-grained indoor location system. In: Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation. pp. 71–84. nsdi'13, USENIX Association, Berkeley, CA, USA (2013), http://dl.acm.org/citation.cfm?id=2482626.2482635

[79] Youssef, M., Agrawala, A.: The horus wlan location determination system. In: Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services. pp. 205–218. MobiSys '05, ACM, New York, NY, USA (2005), http://doi.acm.org/10.1145/1067170.1067193

[80] Yu, M.C., Yu, T., Wang, S.C., Lin, C.J., Chang, E.Y.: Big data small footprint: the design of a low-power classifier for detecting transportation modes. Proceedings of the VLDB Endowment 7(13), 1429–1440 (2014)